Instituto Superior Politécnico Gaya
Escola Superior de Ciência e Tecnologias

Mestrado de Administração de Redes e Sistemas Informáticos

2012/2013

Tese do Projeto

iPortalDoc – Motor de Pesquisa Avançada

Alexandra Sofia dos Santos Rocha nr.1572
assr@ispgaya.pt

Orientador interno: Fernando Almeida
Orientadora externa: Telma Salgueiro

Fevereiro de 2013

Declaração de originalidade e respeito pelos direitos de autor

Alexandra Sofia dos Santos Rocha portador do cartão cidadão 12773960 declara que este trabalho foi por mim realizado na íntegra e é original. Confirmo também que o material proveniente de fontes consultadas está devidamente assinalado e foi referenciado na sua totalidade.

_____

[Alexandra Rocha]

N. de Gaia, 20 de Fevereiro de 2013

"Meu filho, se receberes as Minhas palavras
e guardares os Meus mandamentos dentro do teu coração,
de sorte que o teu ouvido preste atenção à sabedoria
e o teu coração esteja inclinado à prudência,
se tu invocares a sabedoria,
e inclinares o teu coração para a prudência;
E a pesquisares como um tesouro escondido,
então compreenderás o temor de Deus
e chegarás ao conhecimento de Deus.
Porque o Senhor é Quem dá a sabedoria,
e da Sua boca procedem a ciência e a prudência.
Ele reserva a salvação para os justos.
e é como um escudo para os que procedem retamente.
Protege os caminhos dos justos
e dirige os passos dos santos.
Então compreenderás a justiça e a equidade,
A retidão e todos os caminhos que conduzem ao bem.
Se a sabedoria penetrar no teu coração
e a ciência deleitar a tua alma,
e reflexão te guardará e a prudência amparar-te-á
e do homem que fala coisas perversas,
dos que abandonam o caminho reto
e andam por caminhos tenebrosos,
que se alegram em fazer o mal e se regozijam em perversidades
Cujos caminhos são tortuosos
e se extraviam por veredas sinuosas."

Provérbios 2:1,15

Agradecimentos

Primeiramente, agradeço a Deus, porque Dele por Ele e para Ele são todas as coisas. Glorifico a Deus pois Ele é a minha fonte de sabedoria e o meu sustento.

Ao meu orientador Prof. Fernando Almeida, sou profundamente grata pela orientação, por suas contribuições, e pela confiança depositada em mim durante o desenvolvimento deste trabalho.

A empresa que me acolheu IPBrick e a orientadora Telma Salgueiro agradeço a oportunidade.

A todos os professores que durante, o meu mestrado, compartilharam comigo os seus conhecimentos, sem os quais teria sido impossível realizar este trabalho.

A minha tia Gracinda, por me ajudar a fazer a revisão deste trabalho.

Aos meus pais, e à minha avó a quem não posso expressar com palavras a minha gratidão por todo o sacrifício dedicado a mim. Por todo amor e fé em prol do meu estudo.

Resumo

Esta tese tem por objetivo descrever, e contextualizar o trabalho levado a cabo, no âmbito da disciplina de Projeto de Investigação Aplicada e de Desenvolvimento/ Estagio Profissional que consiste no desenvolvimento de um motor de pesquisa avançado para iPortalDoc.

O iPortalDoc é um Sistema de Gestão Documental e Workflow que, para além das funcionalidades tradicionais, deste tipo de software, como o arquivo em formato eletrónico, assegura a eficiência de todos os fluxos de trabalho dentro de uma empresa, determinando quem faz o quê, segundo um procedimentos pré-definido.

Para além disso, o iPortalDoc disponibiliza funcionalidades, como a sincronização com uma central de comunicações, e o acesso a todas as fontes de informação, incorporando assim qualquer tipo de documento. Este gestor documental, permite o acesso de entidades externas, sendo um software, que integra com ERP's e possibilita a assinatura digital, numa solução que se adapta a qualquer empresa, independente da sua dimensão.

Neste projeto foi desenvolvido um motor de pesquisa avançado, que seja eficiente, de fácil utilização, intuitivo, que o utilizador não precise saber SQL, e que seja rápido a dar resposta ao utilizador na sua procura. Este motor de pesquisa foi integrado com o sistema do iPortalDoc – Gestão Documental.

A técnica utilizada para o desenvolvimento do motor de pesquisa é baseado no *full text search*. O princípio desta técnica é examinar todas as palavras, em todos os campos da tabela (base de dados), e tenta corresponder a critérios de pesquisa (por exemplo, palavras fornecidas por um utilizador).

Com essa técnica trouxe para os clientes da empresa uma maior rapidez dos resultados da pesquisa ordenados por ranking. Com os resultados obtidos, por esta técnica, permite à empresa corresponder às necessidades dos seus clientes sendo assim o princípio de um processo de inovação e melhoria para esta.

Abstract

This thesis aims to describe and contextualize the work undertaken within the discipline of Project Development and Applied Research/Internship Training which consists in developing a search engine for advanced iPortalDoc.

The iPortalDoc is Document Management System and Workflow that, in addition to the traditional features, this type of software, such as electronic file format ensures the efficiency of all workflows within an organization, determining who does what, according to pre-defined procedures.

Futhermore, iPortalDoc provides functionalities such as synchronization with a central communication and access to all information sources – thereby incorporating any type of document. This document manager allows access to external entities, and software, which integrates with ERP and enables the digital signature, a solution that adapts to any business, regardless of size.

In this project we developed and advanced search engine, which is efficient, easy to use, intuitive that you do not need to know SQL, and is quick to respond to the user in your search. This search engine was integrated with the system iPortalDoc – Document Management.

The technique used for the development of the search engine is based on full text search. The principle of this technique is that examines all the words in all the fields of the table, database, tries to match search criteria (e.g. word supplied by a user).

With this technique brought to the company's customers a faster search results ordered by ranking. With the results obtained enables the company to meet the needs of its customers and is the beginning of a process of innovation and improvement.

Resumé

Cette thèse vise à décrier et contextualiser la travail réalisé au sein de la discipline de Développement de Projets et la Recherche Appliquée/ Stage de Formation qui consiste à développer un moteur de recherche avancé pour iPortalDoc.

Le iPortalDoc est un Système de Gestion Documentaire et de Workflow que, en plus des caractéristiques traditionnelles, ce type de logiciel, tels que le format de fichier électronique assure l'efficacité de tous les flux de travail au sein d'une organisation, de déterminer qui fait quoi, selon une des procédures prédéfinies.

En outre, iPortalDoc offre des fonctionnalités telles que la synchronisation avec un central de communication et l'accès à toutes les sources d'information, incorporant ainsi tout type de document. Ce gestionnaire de documents permet d'accéder à des entités externes et les logiciels, qui intègre l'ERP et permet la signature numérique, une solution qui s'adapte à toutes les entreprises, indépendamment de leur taile.

Dans ce projet, nous avons développé un moteur de recherche avancé, ce qui est efficace, facile à utiliser, intuitif, que vous n'avez pas besoin de connaître le langage SQL, et est prompt à vous répondre dans votre recherche. Ce moteur de recherche a été intégré au système de iPortalDoc – Gestion des documents.

La technique utilisée pour le développement du moteur de recherche est basé sur la recherche plein texte. Le principe de cette technique est que examine tous les mots dans tous les champs de la table, base de données, tente de faire correspondre les critères de recherche (par exemple, des mots fournis par un utilisateur).

Avec cette technique apportée aux clients de la compagnie résultats d'une recherche plus rapide commandé par le classement. Avec les résultats obtenus permet à l'entreprise de répondre aux besoins de ses clients et c'est le début d'un processus d'innovation et d'amélioration.

Conteúdos

Índice de imagens

Índice Tabelas

Índice de Equações

Abreviaturas

AJAX	*Asynchronous JavaScript and XML*
BD	*Base de Dados*
BIR	*Binary Indepence Retrieval*
CSS	*Cascading Style Sheets*
DOC	*Documento*
FTS	*Full Text Search*
GNU	*General Public License*
HTML	*HyperText Markup Language*
HTTP	*Hyper Text Transfer Protocol*
IR	*Information Retrieval*
ISO	*International Organization for Standardization*
JS	*JavaScript*
PHP	*HyperText Preprocessor*
PR	*Page Rank*
Rec	*Documentos Recuperados*
Rel	*Documento Relevantes*
RI	*Recuperação de Informação*
RR	*Documentos Relevantes e Recuperados*
SGBD	*Sistema de Gestão de Base de Dados*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
WWW	*World Wide Web*

1.Introdução

1.1. Enquadramento

Esta tese é realizada no âmbito da disciplina Estagio/Projeto de Investigação Aplicada, do 2º ano do Mestrado Administração de Redes e Sistemas Informáticos. Nesta disciplina foi realizado estágio de duração de 6 meses na empresa IPBrick, onde foi desenvolvido o projeto iPortalDoc – Pesquisa Avançada.

1.2. Apresentação da empresa

A empresa em que estou a estagiar é a IPBrick, também conhecida como iPortalMais. Nesta parte irei apresentar a empresa.

A iPortalMais nasceu em Maio de 2002 durante uma época de intensa atividade do Linux na cidade do Porto. Nessa altura organizavam muitos eventos de promoção do Linux dentro da Faculdade de Engenharia da Universidade do Porto (FEUP), e em 2000 fez-se o primeiro evento nas ruas da cidade com o patrocínio da Câmara Municipal do Porto e da FEUP. Impulsionada por esta onda Linux, a empresa nasce com várias áreas de intervenção assentes em soluções Linux e sendo a instalação e configuração de servidores Linux nos seus clientes.

Com a crise do mundo na TI a instalar-se a partir dos fins de 2001, esta área torna-se numa parte preponderante da atividade da iPortalMais. Há menos dinheiro para investir em tecnologia e as soluções Linux com um misto de software livre e serviços, são melhores para os clientes e ótimas para os integradores dos serviços.

O primeiro sinal de mudança de atitude, em meado de 2001, a empresa começa a dar os primeiros passos na I&D (Investigação e Desenvolvimento), e começa a conceber uma solução de Gestão Documental com uma abordagem inovadora: a gestão documental como um serviço de rede (networking) que assenta num conjunto de protocolos de comunicação muito utilizados na Internet. A base do Gestor Documental (iPortalDoc) era obviamente baseada em Linux. O projeto evolui com bastante êxito e no fim de 2002 já havia vários parceiros em vista para a comercialização do iPortalDoc. Os problemas começaram quando foi preciso ensinar os parceiros a instalar o Linux para se comercializar a gestão documental. Com gastos muito altos em formação, a iPortalMais decide criar uma ferramenta para instalar automaticamente o servidor Linux da Internet de suporte ao iPortalDoc. O parceiro principal em Gestão Documental, a Peoples Conseil, comunica que realmente a ferramenta poderia ser vendida, e

vende num curto espaço-tempo. A iPortalMais toma consciência que, com aquela ferramenta, o seu departamento I&D havia criado mais um produto. Que foi em seguida batizado de IPBox e pouco depois rebatizado de IPBrick, uma vez que IPBox já existia nas hierarquias ".com".

No ano 2002, as soluções Linux foram criticadas por não terem antivírus. A procura de uma solução de segurança adequada às instalações de servidores Linux terminou com a seleção da Kaspersky antivírus como a solução que resolvem o problema da falta do antivírus. Esse dia foi muito importante para a empresa, pois o distribuidor do produto em Portugal estava na falência, a iPortalMais começa a vender este produto com a IPBrick e em Março de 2003 é convidada para distribuir o Kasperky em Portugal. É a partir daí que a iPortalMais muda radicalmente a sua forma de ser e está no mercado. A pressão do fabricante russo para obter resultados obrigou a distribuir mesmo o produto, a investir em marketing, relações públicas e criar um canal de distribuição. A iPortalMais deixou de ser unicamente um integrador de serviços, fazendo com que o desenvolvimento de tecnologias próprias aliadas à capacidade de distribuição leva-se a um caminho distinto do inicial.

A internacionalização da iPortalMais começa a ganhar algumas raízes na relação com a Kaspersky Lab, pois com o software deles integrado na IPBrick os servidores Linux de instalações automáticas começam a chamar a atenção de fabricantes mundiais e dos seus distribuidores noutros países. Os primeiros parceiros internacionais da iPortalMais são os distribuidores da Kaspersky e o produto começa a ser promovido no estrangeiro. Parte da história da iPortalMais passa sem dúvida pela parceria com a Kaspersky. Assim, a primeira participação numa feira foi impulsionada por Kaspersky, e a iPortalMais surge com grande êxito em Lisboa na COMTEC 2004, logo a seguir, em Janeiro de 2006 em Paris, participa na Solutions Linux. O ponto alto da presença da iPortalMais na feira de Março de 2006 é a sua participação no CeBIT.

1.3. Objetivos

O objetivo é desenvolver um motor de pesquisa avançado para o iPortalDoc otimizado para SGBD Oracle e PostgreSQL. Este motor deverá permitir a obtenção de informação, com base em pesquisas por metadados, palavras ou frases e com a possibilidade de utilização de operadores booleanos, sendo que a pesquisa e a apresentação dos resultados deverá ter em conta rankings.

1.4. Metodologia

Nesta subseção descrevo como será feito o desenvolvimento do projeto, e de uma forma resumida as etapas da metodologia que vou aplicar.

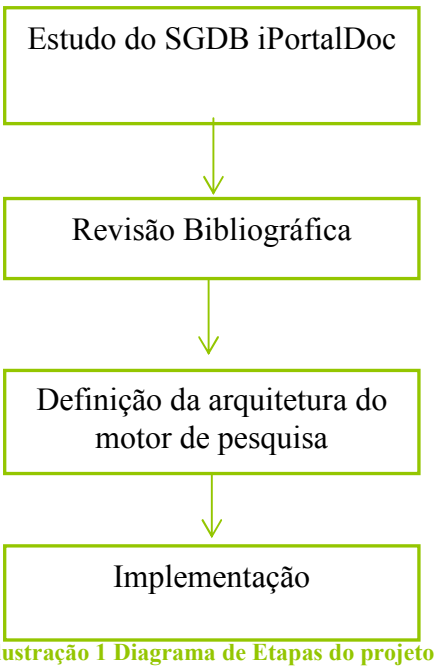

Ilustração 1 Diagrama de Etapas do projeto

Etapa 1: Estudo do SGDW iPortalDoc.

Como funciona o projeto, testar as funcionalidades, realizar um relatório de como funciona o sistema de Gestão de Documental e Workflow.

Etapa 2: Revisão bibliográfica para investigar soluções existentes sobre a pesquisa de palavras-chave em bases de dados.

Realizar um relatório descritivo de soluções para o motor de pesquisa em base de dados.

Etapa 3: Desenvolvimento do motor de pesquisa.

Será realizado um relatório técnico de como o motor de pesquisa foi realizado.

Etapa 4: Implementação do motor de pesquisa.

As ferramentas utilizadas para o desenvolvimento foram o PgAdminIII para a base de dados Postgres, e o Quanta a para linguagem de programação PHP.

Ao ser concluído será implementado nos servidores dos clientes como uma atualização, com as atualizações que estão ser feitas.

1.5. Resultados esperados

Até o final do projeto, espero que compreender a técnica de Full Text Search, e obter uma visão mais abrangente dos modelos clássicos. Pretende-o alcançar os objetivos descritos no item *objetivo*. Além disso espero que o protótipo desenvolvido possa demonstrar as boas práticas e inovações possíveis.

1.6. Estrutura da dissertação

Este documento é composto por 8 capítulos. O **primeiro capítulo** faz-se a introdução onde é descrita a contextualização, motivação e objetivos da dissertação.

O **segundo capítulo** refere-se à base teórica desta matéria, descrevem-se as áreas de pesquisas (modelo clássico, recuperação de informação e full text search).

O **terceiro capítulo** refere-se a requisitos funcionais e não funcionais do projeto.

O **quatro capítulo** refere a descrição e o desenvolvimento do projeto do motor de pesquisa, tentando com isso um motor de pesquisa simples e robusto que consiga extrair os termos dos documentos.

O **quinto capítulo** refere a estrutura do tempo utilizado para a realização do projeto.

O **sexto capítulo** refere os meios utilizados neste projeto, quanto a recursos técnicos, humanos e equipamentos.

O **sétimo capítulo** refere problemas surgidos e decisões, tomadas ao longo do projeto.

O **oitavo capítulo** refere os testes executados e os seus resultados. Expõe em que consistiam os testes feitos e protocolos usados bem como as propriedades dos mesmos. Refere ainda os resultados e se estes estão enquadrados com o esperado ou não.

O **nono capítulo** apresenta as conclusões da dissertação assim como possíveis trabalhos futuros.

2. Bases teóricas

2.1. Recuperação de Informação

Recuperação de informação (RI) de acordo com Salton (Salton G. , 1968) "é a área de pesquisa que se preocupa com a estrutura, análise, organização, armazenamento, recuperação e procura de informação". A representação e a organização desta informação devem permitir que os utilizadores tenham fácil e rápido acesso à informação pretendida. (Kowalski, 1997) cita que o principal objetivo de um sistema RI é diminuir a dificuldade do utilizador em localizar a informação requisitada. Por essa dificuldade, pode-se entender que o tempo gasto num utilizador são todos os passos necessários a um processo de pesquisa até que a informação requerida possa ser acedida.

Baeza-Yate e Ribeiro-Neto (Baeza-Yates & Ribeiro-Neto, 1999) afirmam que "recuperação efetiva de informação importante é diretamente afetada pela tarefa do utilizador e pela visão lógica dos documentos adotada por um sistema de recuperação". A tarefa do utilizador no âmbito RI significa exprimir as necessidades dele em consultas que possam ser entendidas por sistema de RI. A visão lógica dos documentos é representada de forma a permitir que as pesquisas sejam possíveis. Normalmente, os documentos são representados por meio de índices formados por palavras, mais conhecidas como termos no âmbito da RI, que compõem esses documentos (Rijisbergen, 1979).

A elaboração de um sistema RI inicia-se, primeiramente, com significado da fonte de informação, isto é, a relação de documentos a serem recuperados. Também é necessário determinar quais são as operações que podem ser realizadas durante um processo de pesquisa. Com estas definições, um desenvolvedor precisa de estabelecer qual é a estrutura necessária ou visão lógica dos documentos, que possibilite tais operações. Definida a visão lógica, o próximo passo é construir um índice que guarde os termos que contém os documentos. Segundo Baeza-Yates e Ribeiro-Neto (Baeza-Yates & Ribeiro-Neto, 1999) "um índice é uma estrutura de dados crítica porque permite rápida procura sobre grandes volumes de dados". A estrutura do índice também é definida pelo desenvolvedor, que analisa as operações a serem executadas sobre o índice, quais as informações que podem ser pesquisadas, com as informações que são recuperadas. Com o índice criado um processo restaurado pode ser começado. Essencialmente este processo começa com o utilizador a descrever a necessidade por meio de consulta, traduzindo as possíveis operações em linguagem que possa ser aplicada sobre o índice. O resultado da aplicação da consulta sobre o índice faz com que a lista de documentos seja ordenada de acordo com o critério, também, definido pelo desenvolvedor para ser apresentado

ao utilizador.

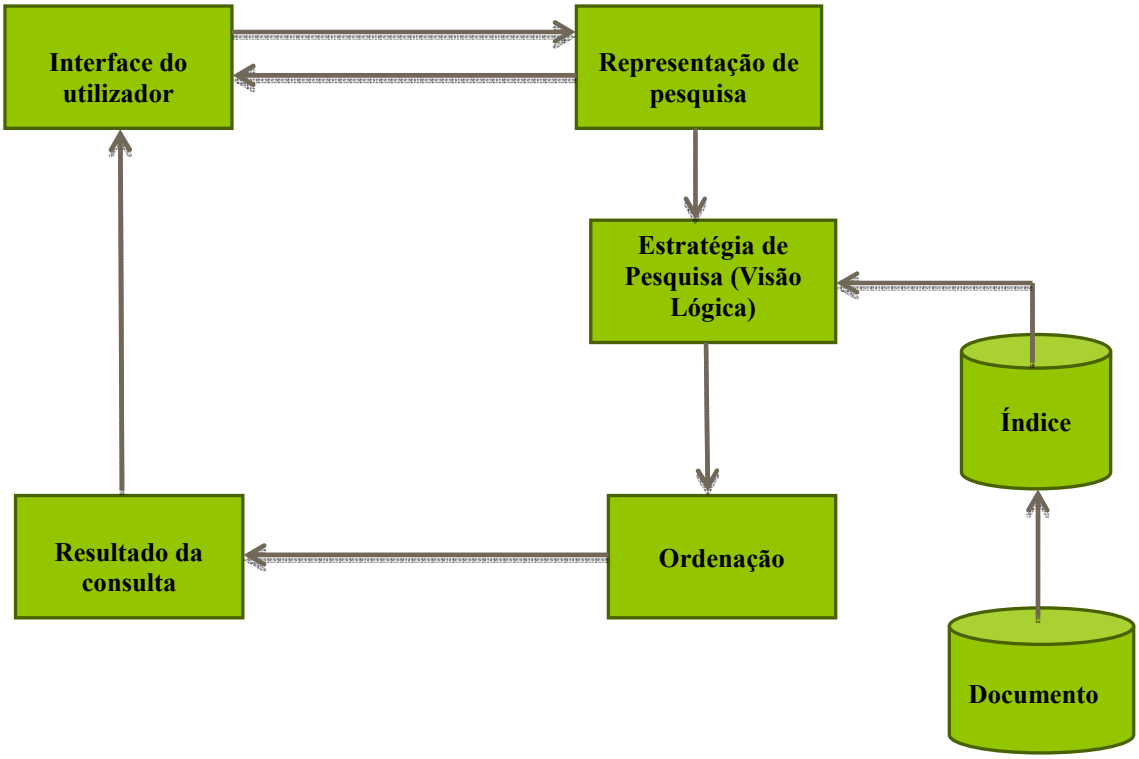

Ilustração 2 Modelo básico de representa de informação. Adaptado de (Ricarte & Gromide, 2004)

A maior parte das pesquisas e do desenvolvimento de RI tem como objetivo e a melhoria da eficácia bem como a competência da tarefa de recuperação (Rijisbergen, 1979). A competência é normalmente medida em termos computacionais (por exemplo performance, tempo de CPU etc.) enquanto a eficácia mesmo sendo subjetiva do ponto de vista do utilizador (Kowalski, 1997) comum, é calculada por meio das medidas *precision* e *recall* (Rijisbergen, 1979) (Baeza-Yates & Ribeiro-Neto, 1999) (Witten, Moffat, & Bell, 1999). *Precision* são a relação entre o número de documento importantes recuperados do número total de documentos importantes. *Recall* é a relação do número de documentos importantes recuperados sobre o número total de documentos importantes. Essas medidas são usadas para avaliar o desempenho de um modelo de RI e para fazer a comparação do desempenho entre diferentes modelos. Na subseção a seguir descreve-se os principais modelos clássicos de RI também conhecidos.

2.2. Modelos de recuperação de informação

Modelos de recuperação de informação têm sido a base de RI desde 1960. De acordo com Baeza-Yates e Ribeiro-Neto (Baeza-Yates & Ribeiro-Neto, 1999), um modelo de RI é composto por:

a. Um conjunto de visões lógicas ou representações de documentos numa coleção;

b. Um conjunto de visões lógicas ou representações da informação requerida por o utilizador;

c. Um framework para reproduzir documentos, consultas e suas relações;

d. Uma função de ordenação que associa um número com uma consulta e um documento.

Diferentes modelos foram e continuam a ser desenvolvidos, e descrevem aspetos inesperáveis à tarefa de recuperação, tais como: estrutura e conteúdo do documento, consultas das necessidades do utilizador e contexto no qual esta tarefa está introduzida (Allan, et al., 2002). Há uma variedade de modelos de RI (Korfhage, 1997) que oferecem diferentes estruturas e linguagens de pesquisa, e que disponibilizam especificidades que podem ser usadas em adaptar à informação armazenada. Segundo Greengrass (Greengrass, 2000), há duas categorias principais de modelos de RI: semânticos e estatísticos. Os modelos semânticos pretendem promover a análise semântica e sintática no intuito de "entender" um texto descrito e linguagem natural. Os modelos estatísticos conferem medidas estatísticas que se referem à comparação entre uma consulta e um documento.

A seguir são apresentados os modelos tradicionais de RI.

2.2.1. Modelo Booleano

Este modelo é baseado na teoria dos conjuntos e na álgebra booleano (Korfhage, 1997) (Baeza-Yates & Ribeiro-Neto, 1999). As consultas são expostas por meio de expressões booleanas utilizando operadores lógicos AND, OR e NOT. Os documentos são representados por meio de um conjunto de termos indexados. A recuperação de um documento acontece se este responder verdadeiramente a uma expressão booleana ou seja a uma consulta. Como o resultado da recuperação de um documento é um valor binário, esse modelo não oferece um mecanismo de ordenação.

A principal vantagem do modelo booleano refere-se à simplicidade. A principal desvantagem é a comparação exata entre consulta e documento, que pode levar à recuperação de poucos ou muitos documentos (Baeza-Yates & Ribeiro-Neto, 1999).

2.2.1.1. AND

Uma expressão conjuntiva de demonstrar $\mathbf{t_1}$ **AND** $\mathbf{t_2}$ recuperará documentos indexados por ambos os termos ($\mathbf{t_1}$ e $\mathbf{t_1}$). Esta operação equivale ao cruzamento dos conjuntos dos documentos indexados pelo termo $\mathbf{t_1}$ com o conjunto dos documentos indexados pelo termo t_2 representado

pela área cinzenta na ilustração 3.

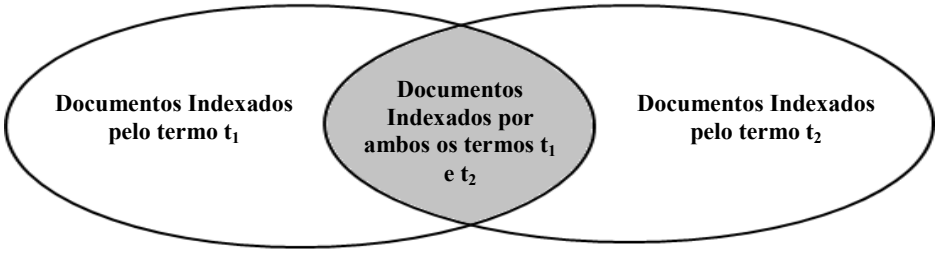

Ilustração 3 Representação do resultado de uma expressão booleano conjuntivo (AND)

2.2.1.2. OR

Uma expressão disjuntiva de exprimir **t_1 OR t_2** recuperará documentos indexados pelo termo **t_1**. Esta operação equivale à união dos conjuntos de documentos indexados pelo termo **t_1** com o conjunto dos documentos indexados pelo termo **t_2** representado pela ilustração 4.

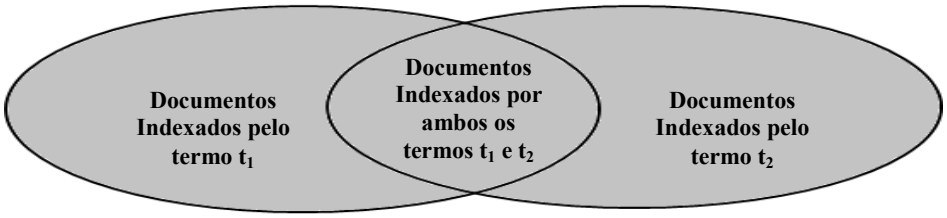

Ilustração 4 Representação do resultado de uma expressão booleano disjuntivo (OR)

2.2.1.3. NOT

Uma expressão que utiliza apenas um termo **t_1** terá como resultado o conjunto de documentos indexados por **t_1**. A expressão **NOT** recuperará os documentos que não são indexados pelo termo **t_1**, representados pela área cinza da ilustração 5.

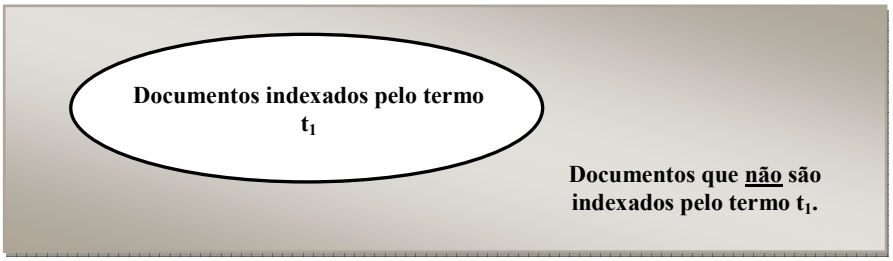

Ilustração 5 Resultado de uma pesquisa negativa (NOT)

A expressão **t_1 NOT t_2** terão o mesmo resultado o conjunto dos documentos indexados por **t_1** e os que não são indexados por **t_2**. Neste caso o operador NOT pode ser visto como um operador de diferença entre conjuntos. Assim, a área cinza da ilustração 6 representa o conjunto dos documentos indexados pelos termos **t_1** menos o conjunto dos documentos por **t_2**.

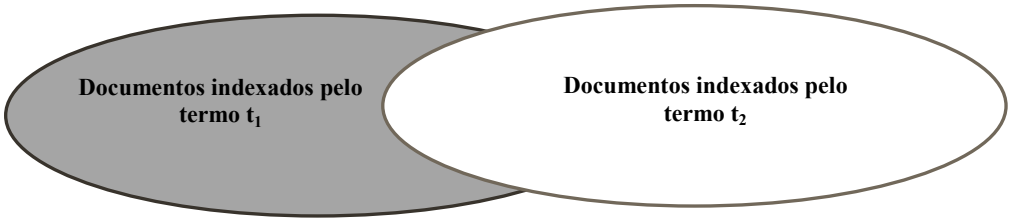

Ilustração 6 Resultado de uma pesquisa booleana com o operado NOT

2.2.2. Modelo booleano estendido

O modelo booleano estendido distingue-se do modelo booleano por usar diferentes operadores e por promover uma função de ordenação (Baeza-Yates & Ribeiro-Neto, 1999) (Lee, 1994) (Greengrass, 2000). Esse modelo booleano atribui o valor um (1) aos termos de uma pesquisa, de acordo com a expressão lógica que estão nos documentos e zero (0) caso contrário. Já o modelo estendido atribui valores que variam de zero (0) a um (1), os quais correspondem a uma estimativa de comparação de uma expressão lógica com um documento. Lee (Lee, 1994) cita que modelos booleanos estendidos (por exemplo conjuntos difusos Waller-Kraft, paice, P-Norm e Infinit-One) têm sido propostos a oferecerem cálculos semelhantes mais robustos entre uma pesquisa e um documento.

A similaridade entre um documento $d_i = (w_{1i}, w_{2i})$ e uma consulta $q = t_1 \; or \; t_2$ é calculada através da equação 1, onde w_{1i} *e* w_{2i} representam os pesos de cada um dos termos de indexação do documento.

$$sim\,(q_{or}, d) = \sqrt{\frac{w_{1i}^2 + w_{2i}^2}{2}}$$

Equação 1 Equação similaridade para o modelo booleano estendido

2.2.3. Modelo Vetorial

No modelo vetorial cada documento é representado por um vetor ou por uma lista de termos ordenados, por exemplo, pela frequência do termo no documento, em vez de apenas um conjunto de termos como utiliza o modelo booleano (Kowalski, 1997) (Baeza-Yates & Ribeiro-Neto, 1999) (Greengrass, 2000) (Salton, Wong, & Yang, A vector space model for automatic indexing, 1975). Cada vetor descreve a posição do documento em um espaço multidimensional, onde cada termo de indexação representa uma dimensão ou eixo. Cada elemento do vetor é normalizado de forma a assumir valores entre zero (0) e um (1). Os pesos mais próximos de um (1) indicam termos com maior importância para a caraterística do documento. A ilustração 7

apresenta a representação gráfica de um documento DOC_1 com termos de indexação t_1 e t_2 pesos 0,30 e 0,5 respetivamente.

	t_1	t_2
DOC_1	**0.3**	**0.5**

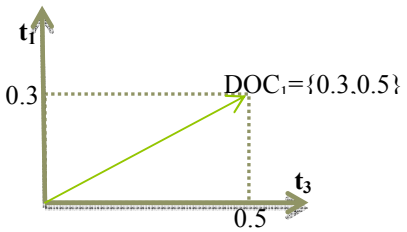

Ilustração 7 Representação vetorial de um documento com dois termos de indexação

Na ilustração 8 mostra os dois documentos DOC_1 e DOC_2 representados num mesmo espaço vetorial. Os números positivos representam os pesos dos seus respetivos termos. Termos que não estão presentes num determinado documento possuem peso igual a zero.

	t_1	t_2	t_3
DOC_1	**0.3**	**0.0**	**0.5**
DOC_2	**0.5**	0.4	0.3

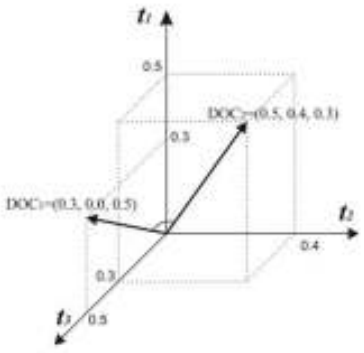

Ilustração 8 Espaço vetorial contendo dois documentos

Da mesma forma os documentos, no modelo vetorial, uma expressão de pesquisa também é representado por um vetor numérico onde cada elemento representa a importância do respetivo termo na expressão de pesquisa. A ilustração 9 mostra a representação da expressão de pesquisa e eBUSCA =(0.2,0.35, 0.1) juntamente com os documentos DOC_1 e DOC_2 num espaço vetorial formado pelos t_1, t_2 e t_3.

	t_1	t_2	t_3
$eBUSCA_1$	0.2	0.35	0.1

	t_1	t_2	t_3
DOC_1	0.3	0.0	0.5
DOC_2	0.5	0.4	0.3

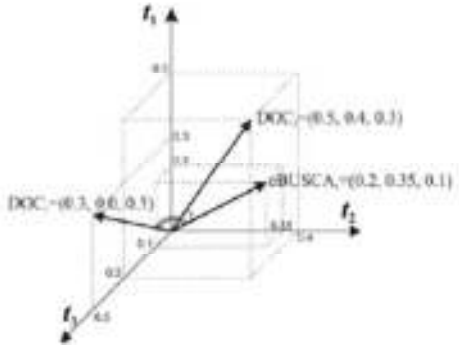

Ilustração 9 Representação de uma expressão de pesquisa em um espaço vetorial

Para que fosse possível apresentar visualmente um espaço vetorial contendo documentos e expressões de pesquisa, nos exemplos acima foram utilizados apenas três termos de indexação na representação dos documentos. Claramente, um sistema real contém um grande número de

termos de indexação e documentos. Um *corpus* contendo um número indefinido de documentos e termos de indexação pode ser representado através de uma matriz onde cada linha representa um documento e cada representa a associação de um determinado termo aos diversos documentos. Um *corpus* contendo documento por exemplo termos de indexação pode ser representando da seguinte forma:

	t_1	t_2	t_3	...	t_i
DOC_1	$w_{1,1}$	$w_{2,1}$	$w_{3,1}$	...	$w_{i,1}$
DOC_2	$w_{1,2}$	$w_{2,2}$	$w_{3,2}$	...	$w_{i,2}$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
DOC_n	$w_{1,n}$	$w_{2,n}$	$w_{3,n}$	...	w_{in}

Ilustração 10 Corpus contendo n documentos e i termos de indexação

2.2.3.1. Cálculo da similaridade

A utilização de uma mesma representação tanto para os documentos como para as expressões de pesquisa permite calcular o grau de similaridade entre dois documentos ou entre uma expressão e cada um dos documentos do *corpus*. Num espaço vetorial contendo ***N*** dimensões a similaridade (***sim***) entre dois vetores ***j*** e ***q*** é calculada através do co-seno do angulo formado por estes vetores, utilizando a seguinte formula.

$$sim(d_j, q) = \frac{\sum_{i=1}^{N} (w_{i,j} \times w_{i,q})}{\sqrt{\sum_{i=1}^{N} w^2_{i,j}} \times \sqrt{\sum_{i=1}^{N} w^2_{i,q}}}$$

Equação 2 Formula para calcular a similaridade

Onde $w_{i,j}$ é o peso do i-ésimo elemento do vetor ***j*** e $w_{i,q}$ é o peso do i-ésimo elemento do vetor ***q***.

O grau de similaridade entre o **DOC_1** e o **DOC_2**, representados na ilustração 8, é calculado o grau de similaridade entre os documentos e é igual a 0.73 ou 73%.

Utilizando a mesma fórmula, pode-se calcular a similaridade entre a expressão **$eBUSCA_1$** e cada um dos documentos **DOC_1** e **DOC_2**, representados na ilustração 9.

$$sim\ (DOC_1\ ,\ eBUSCA_1\ ) = 0.45 = 45\%$$

$$sim\ (DOC_2\ ,\ eBUSCA_1\ ) = 0.95 = 95\%$$

Portanto a expressão eBUSCA1 possui um grau de similaridade de 45% com o documento **DOC_1** e de 95% com o **DOC_2**.

Os valores de similaridade entre uma expressão de pesquisa e cada documento do *corpus* são utilizados na ordenação dos documentos resultantes. Portanto, no modelo vetorial o resultado de uma pesquisa é um conjunto de documentos ordenados pelo grau de similaridade entre cada documento e a expressão de pesquisa.

Uma característica do modelo vetorial é que os termos de indexação são independentes, isto é, não são considerados relacionamentos existentes entre eles. Embora alguns autores apontem essa característica como uma desvantagem, para Baeza-Yate e Reibeiro-Neto (Baeza-Yates & Ribeiro-Neto, 1999) não há evidências conclusivas que apontem que tais dependências afetem significativamente o desempenho de um sistema de recuperação de informação. Uma importante limitação do modelo é não permitir formulação de pesquisas booleanas o que restringe consideravelmente.

2.2.4. Modelo Probabilístico

O modelo probabilístico proposto por Robertson e Jones (Robertson & Jones, 1976) posteriormente conhecido como Binary Indepence Retrieval (Baeza-Yates & Ribeiro-Neto, 1999), tenta revelar o processo de recuperação de informação sob um ponto vista probabilístico.

Os modelos probabilísticos trabalham com um conjunto *Q* de consultas e um conjunto *D* de documentos de uma coleção (Fuhr & Pfeifer, 1994). Na maioria dos modelos de recuperação de informação as pesquisas e os documentos são representados por palavras-chave, que podem ser extraídas de modo manual ou automática. Essas palavras-chave são representadas como um vetor onde cada elemento corresponde a um termo.

Segundo Crestani et al. (Crestani, Lalmas, Rijsbergen, & Campbell, 1998), a teoria probabilística é um caminho para tratar essa incerteza na recuperação.

Dada uma expressão de pesquisa, pode-se dividir o *corpus* (com N documentos) em quatro subconjuntos diferentes (ilustração 11).

- ✓ O conjunto dos documentos relevantes – **Rel**;
- ✓ O conjunto dos documentos recuperados – **Rec**;
- ✓ O conjunto dos documentos relevantes que foram recuperados – **RR**;
- ✓ E o conjunto dos documentos não relevantes e não recuperados. - $\overline{Rel}$ e $\overline{RR}$

O conjunto dos documentos relevantes e recuperados (***RR***) é resultante da interseção dos conjuntos ***Rel*** e ***Rec***.

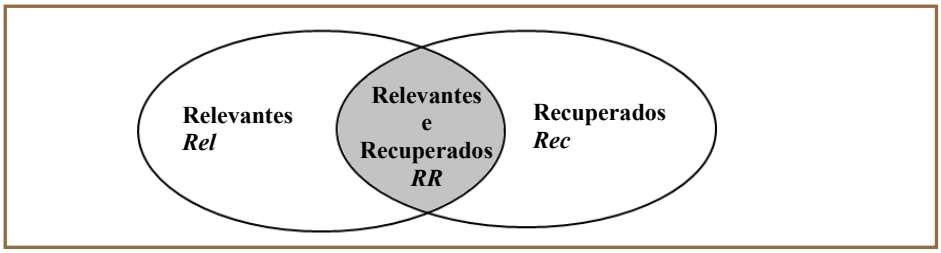

Ilustração 11 Subconjuntos de documentos após a execução de uma pesquisa

O resultado ideal de uma pesquisa é o conjunto que conte apenas os documentos relevantes para o utilizador, isto é, todo o conjunto *Rel*. Se o utilizador conhece exatamente o que distingue os documentos desse conjunto dos de mais conjuntos do *corpus* será fácil recuperá-los. No entanto, como as características dos documentos não são conhecidas, tenta-se adivinhar tais características através da formulação de uma expressão de pesquisa, gerando uma primeira descrição probabilística desse conjunto. Com os resultados obtidos após a execução da primeira pesquisa é possível melhorar os resultados através de instruções com o utilizador.

Seja *Rel* o conjunto dos documentos relevantes e $\overline{Rel}$ o complemento de *Rel*, ou seja, o conjunto dos documentos não relevantes. A probabilidade ***d*** ser relevante em relação à expressão de pesquisa é representado por $p(Rel|d)$. A probabilidade de um documento ser considerado não relevante é representada por $p(\overline{Rel}|d)$. A similaridade (***sim***) de um documento **d** em relação à expressão de pesquisa e**Busca** é definida como:

$$sim(d, eBUSCA) = \frac{p(Rel|d)}{p(Rel|d)}$$

Utilizando o teorema Bayes obtêm-se a seguinte expressão:

$$sim(d, eBUSCA) = \frac{p(d|Rel) * p(Rel)}{p(d|\overline{Rel}) * p(\overline{Rel})}$$

A expressão $p(d|Rel)$ representa a probabilidade de se selecionar o documento **d** do conjunto dos documentos relevantes *Rel* e $p(d|\overline{Rel})$ representa a probabilidade de se selecionar o documento **d** do conjunto dos documentos não relevantes. A expressão $p(Rel)$ representa a probabilidade de um documento selecionado aleatório ser relevante, enquanto $p(\overline{Rel})$ representa a probabilidade de um documento não relevante.

Considerando $p(Rel)$ e $p(\overline{Rel})$ sejam iguais para todos os documentos do *corpus* a fórmula da similaridade pode então ser escrita como:

$$sim(d, eBUSCA) = \frac{p(d|Rel)}{p(d|Rel)}$$

Um documento é representado por um vetor binário cuja presença e a ausência de um determinado termo de indexação (t_i) é designado respetivamente por 1 ou 0.

	t_1	t_2	t_3	...	t_n
Doc	w_1	w_2	w_3	...	w_n

Ilustração 12 Representação de um documento indexado

Onde w_i pode assumir o valor zero (0) ou um (1) indicando a ausência ou a presença do termo de indexação t_i no conjunto dos indexadores do documento *Doc*.

A probabilidade de um termo t_i estar presente num documento selecionado do conjunto *Rel* é representado por $p(t_i|Rel)$ e $p(\overline{ti}|Rel) = 1$, e ignorando fatores que são constantes para todos os documentos no contexto de uma mesma pesquisa tem-se finalmente:

$$sim(d, eBUSCa) \approx \sum_{i=1}^{t} \left(\log \frac{p(ti|Rel) * p(\overline{ti}|\overline{Rel})}{p(ti|\overline{Rel}) * p(\overline{ti}|Rel)} \right)$$

Equação 3 Formula da similaridade

Esta expressão é fundamental para ordenar os documentos no modelo probabilístico. Todo o cálculo da probabilidade resume-se a um problema de contagem. Portanto para uma determinada expressão de pesquisa, pode-se representar os documentos da seguinte forma:

	Relevante	**Não-Relevante**	
Documento contendo t_i	R	n-r	N
Documento que não contem t_i	R-r	N-R-n+r	N-n
	R	N-R	N

Tabela 1 Tabela de contagem (Harman)

Considerando um *corpus* com *N* documentos e um determinado termo t_i, existe no *corpus* um total de *n* documentos indexados por t_i. Desses *n* documentos apenas *r* são relevantes.

No início do processo da pesquisa não se sabe qual é o conjunto de documentos relevantes (R), já que nenhum documento foi ainda recuperado. Portanto, antes da primeira pesquisa é necessário fazer algumas simplificações tais como:

a) Assumir que $p(t_i|Rel)$ é constante e igual a 0.5 para todos os termos t_i;

b) Assumir que a distribuição dos termos de indexação dos documentos é uniforme.

Assim obtêm-se a seguinte fórmula:

$$sim(d, eBUSCA) \approx \sum_{i=i}^{t} \left(\log \frac{N-n}{n} \right)$$

Através dessa fórmula é ordenado o conjunto de documentos, resultado da primeira pesquisa. Contendo esse conjunto de documentos, o utilizador seleciona alguns documentos que considera relevantes para a sua necessidade. O sistema então utiliza esta informação para tentar melhor os resultados seguintes.

Para exemplificar, será considerado um corpus contendo 6 documentos e 10 termos de indexação.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Doc_1	**1**	0	0	**1**	0	0	0	**1**	**1**	0
Doc_2	0	0	0	0	0	0	0	**1**	**1**	**1**
Doc_3	0	1	0	0	0	**1**	**1**	0	0	0
Doc_4	**1**	0	0	**1**	0	0	0	0	0	**1**
Doc_5	0	0	0	0	0	0	0	**1**	**1**	0
Doc_6	0	0	**1**	0	**1**	0	0	0	0	0

Tabela 2 Exemplificam de vários documentos indexados

A expressão de pesquisa (*eBusca*) será composta pelos termos t_4 e t_{10} sendo representada pelo seguinte vetor:

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
eBUSCA	0	0	0	1	0	0	0	0	0	1

Tabela 3 Representação da expressão de pesquisa eBusca

Após a execução da primeira pesquisa os documentos recuperados serão apresentados em ordem do valor resultado da equação 2 aplicada a cada documento. Alguns documentos, como no caso dos documentos 3, 5, e 6, não são recuperados pois apresentaram valor menor ou igual a zero.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	sim(DOC_i, eBUSCA)
DOC_4	**1**	0	0	**1**	0	0	0	0	0	**1**	0.51
DOC_1	**1**	0	0	**1**	0	0	0	**1**	**1**	0	0.26
DOC_2	0	0	0	0	0	0	0	**1**	**1**	**1**	0.26

Tabela 4 Resultado da aplicação da fórmula da similaridade

Com este primeiro resultado o utilizador poderá selecionar alguns documentos que são úteis. No exemplo apenas três documentos resultaram da primeira pesquisa. Porém, se na pesquisa resultar uma quantidade muito maior de documento basta selecionar alguns documentos que considerasse relevantes. No exemplo, o documento Doc_1, apesar de ter o mesmo grau de similaridade do documento Doc_2 ele não foi considerado relevante pelo utilizador. Após submeter novamente a expressão de pesquisa, juntamente com os documentos selecionados o sistema calculará para cada documento um valor de similaridade utilizando a equação 2. Esse valor será utilizado para ordenar o conjunto de documentos recuperados.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	sim(DOC_i, eBUSCA)
DOC_4	**1**	0	0	**1**	0	0	0	0	0	**1**	2.02
DOC_2	0	0	0	0	0	0	0	**1**	**1**	**1**	1.65
DOC_1	0	0	0	0	0	0	0	**1**	**1**	0	0.37

Tabela 5 Resultado da aplicação da fórmula da similaridade pela segunda vez

Com a repetição desse processo espera-se uma melhoria progressiva nos resultados. O utilizador poderá repetir esse processo de seleção dos documentos relevantes até que o conjunto de documento recuperados satisfaça a sua necessidade de informação.

A principal virtude do modelo probabilístico está em reconhecer que a atribuição de relevância é uma tarefa do utilizador. É o único modelo que incorpora explicitamente o processo de *Relvance Feedback* como base para a sua operacionalização. Segundo Saltone McGill (Salton & McGill, Introduction to Modern Information Retrieval, 1983), na prática só os documentos melhores classificados são examinados, a ideia principal consiste em selecionar termos importantes, ou expressões, dos documentos que são identificados como relevantes pelo utilizador, esse processo aumenta a importância desses termos numa nova formulação de consulta. Como resultado, uma nova consulta, está será direcionada para os documentos relevantes e não serão verificados os não relevantes.

É importante observar que o modelo probabilístico pode ser facilmente implementado utilizando a estrutura proposta pelo modelo vetorial, permitindo integrar as vantagens desses dois modelos em um sistema de recuperação de informação.

Embora o modelo probabilístico tenha uma forte base teórica, nas hipóteses assumidas para realizar simplificações nos cálculos probabilísticos podem deixar dúvidas sobre sua precisão. Uma simplificação bastante questionável está no facto de o modelo considerar os pesos dos termos de indexação como sendo binário, ou seja, no modelo probabilístico não é considerada a frequência com que os termos ocorrem no texto dos documentos.

Algumas experiências, utilizando poucos documentos demonstram que este modelo produz resultados pouco superiores em relação ao modelo booleano. Pode ser que no contexto heterogéneo e complexo da web os métodos probabilísticos venham a se destacar. Porem a sua complexidade desanima muitos desenvolvedores de sistemas o que faz com que abandonem o modelo booleano e vetorial (Robertson & Jones, 1976).

2.3. Full Text Search

O *full text search* (FTS) é uma técnica de pesquisa de textos em base de dados, o que torna possível o processamento e indexação de documentos para uma pesquisa mais rápida.
Em FTS o motor de pesquisa examina todas as palavras em todos os documentos armazenados, bem como, tenta corresponder a critérios de pesquisa, por exemplo, por palavra-chave fornecida pelo utilizador. A técnica de FTS on-line tornou-se comum em base de dados bibliográficos. Muitos sites e programas aplicativos, por exemplo processamento de texto, fornecem recursos do FTS.

2.3.1. Indexação

Ao lidar com um pequeno número de documentos é possível o motor de pesquisa de FTS para digitalizar diretamente o conteúdo dos documentos com cada *query*, uma estratégia chamada digitalização em série. Isto é o que algumas ferramentas rudimentares fazem como o *grep* quando pesquisa.

No entanto, quando o número de documentos para a pesquisa é grande ou a quantidade de consulta de pesquisa para executar é substancial, o problema da pesquisa FTS é regularmente divida em duas tarefas:

1º. Indexação irá analisar o texto de todos os documentos e construir uma lista de termos de pesquisa, conhecida como índice;
2º. Pesquisa, ao realizar uma consulta específica, apenas o índice é referenciado em vez do texto dos documentos originais.

O *indexer* fará uma entrada no índice para cada termo ou palavra encontrada num documento e, possivelmente, a sua posição relativa dentro do documento. Normalmente, o índex irá ignorar *stop word*, como por exemplo na língua inglesa o "the", que são ambos muito comuns e carregam um significado muito pouco para ser útil para a pesquisa, o mesmo acontece na língua portuguesa com os artigos (a, e, as, os). Alguns *indexer* também empregam a linguagem específica decorrente das palavras a ser indexadas, como por exemplo, qualquer das palavras "drive", "drove", "driven" será gravado no índice sob o conceito de "drive" uma única palavra.

2.3.2. Page Rank

Page Rank é um algoritmo de análise de links, é utilizado pela ferramenta de pesquisa da internet Google, que atribui um peso numérico a cada elemento de um conjunto de *hiperlinks* de documentos, tais como a www, com o propósito de "medir" a sua importância relativa dentro do conjunto (Google, 2013). O algoritmo pode ser aplicado a qualquer coleção de entidades com citações e referências reciproca. O peso numérico que ele atribui a qualquer determinado elemento *E* é referido como *Page Rank* de *E* e denotado pelo *PE(E)*.

O *Page Rank* (Page, Brin, Motwani, & Winograd, 1998) (Franceschet, 2011) resulta de um algoritmo matemático baseado no grafo, o *webgraph*, criado por todas as páginas www como nós e *hiperlinkes* como bordas. O valor da classificação indica a importância de uma página específica. Um *hiperlink* para uma página conta como um voto a favor para aquela página. O *page rank* de uma página é definido recursivamente e depende do número e do valor do *page*

rank de todas as páginas que apontam para ela, chamados de "incoming links" (links recebidos). Uma página que está ligada a muitas páginas com *page rank* alto recebe uma alta classificação para si. Se não há links para uma página web então não há suporte para esta página.

2.3.2.1. Algoritmo

O *page rank* é uma distribuição de probabilidade utilizada para representar a probabilidade de uma pessoa que aleatoriamente clica em *links* que chegam a qualquer página particular. O *page rank* pode ser calculado para coleções de documentos de qualquer tamanho. Supõe-se que em diversas pesquisas que a distribuição é dividida igualmente entre todos os documentos da coleção no início do processo computacional. Os cálculos do *page rank* exigem que variem passagens chamadas de iterações, através da coleção, para ajustar os valores aproximados do *page rank* melhor refletindo o valor teórico da verdade.

$$PR(P_0) = p * \sum_{i=1}^{n} \frac{PR(P_i)}{c(P_i)} + \frac{1-p}{n}$$

- ✓ $PR(P_0)$ é o *PageRank* da página P_0
- ✓ $PR(P_i)$ é o *PageRank* da página P_9 que liga à página P_0
- ✓ $C(P_i)$ é o número de ligações de saída ("outbound links" ou "out-degree") na página P_i;
- ✓ $C(P_i)$ deve ser 0
- ✓ n é o numero de paginas
- ✓ p é um fator, 0<p<1;

O *page rank* forma uma distribuição de probabilidade sobre as páginas web, sendo que a soma dos *page rank* de todas as páginas web é 1. Na verdade no seu papel original "The antomy of a Large-Scale Hypertextual Web Search Engine", Lawrence Page e Sergey Brin definiram page rank por:

$$PR(P_0) = p * \sum_{i=1}^{n} \frac{PR(P_i)}{c(P_i)} + (1-p)$$

Nesta versão a probabilidade de um passeio imprevisível de alcançar a página é ponderada pelo número total de páginas. Neste caso, a soma dos page rank de todas as páginas da web é *n*. Esta variante é muitas vezes usada para ilustrar o modo de funcionamento do algoritmo pois não necessita do valor de *n*.

3. Requisitos

3.1. Requisitos Funcionais

Os requisitos técnicos do sistema são os seguintes

- ✓ Gestão Documental
  - Permitir a inserção das informações dos documentos – já existia esta opção no sistema desenvolvido, foi alterado para permitir que se guarda como termos;
  - Permitir a modificação das informações de documentos – a mesma maneira que na inserção;
  - Permitir a eliminação das informações de documentos - essa opção já existia foi alterada para ser eliminado também os termos. Elimina tudo o que faz parte do documento que deseja eliminar;
- ✓ Pesquisar documentos
  - Possibilidade de visualização dos documentos;
  - Possibilidade de download dos documentos pesquisados;

3.1.1. Modelo Caso de Uso

Durante todos os processos existente vários atores que têm o papel específico de interação nas várias fases processuais do sistema. Esses atores são os seguintes:

☙ Utilizador – todo e qualquer utilizador que faça uso da aplicação na sua vertente operacional de negócio. É necessário autenticar-se no sistema para ter acesso às funcionalidades específicas que lhe estão atribuídas pelo administrador;

☙ Administrador – tem por função administrar o sistema. Esta função inclui todas as tarefas de administração comuns de aplicações informáticas.

A ilustração seguinte ilustra os processos gerais que o utilizador deve fazer: a introdução, modificação e eliminação das informações dos documentos, como a pesquisados documentos.

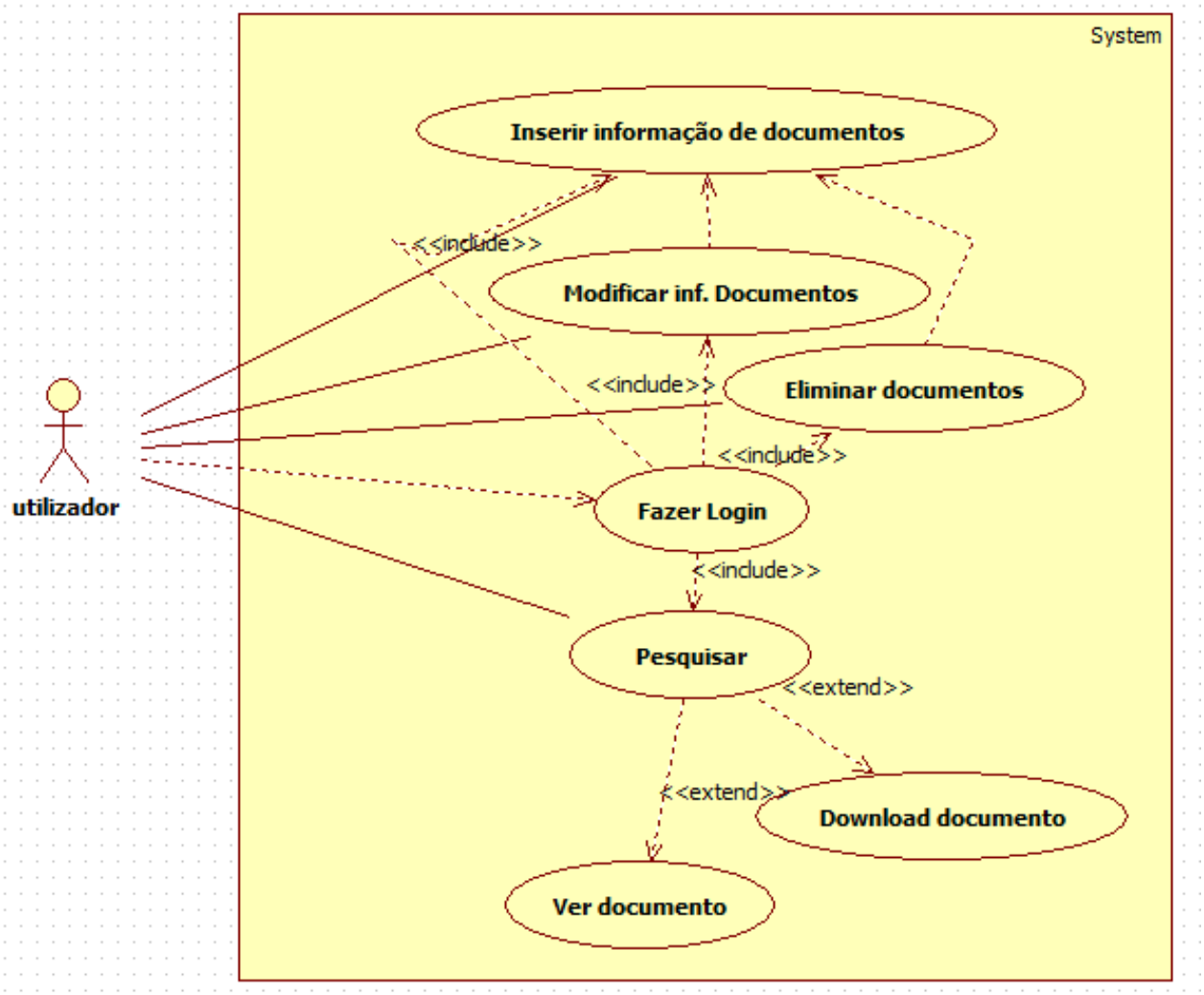

Ilustração 13 Caso de Uso – Utilizador

De seguida descrevo as tabelas de caso de uso, o que cada caso uso consiste qual o seu ator, se sua pré-condição e pós-condição para poder funcionar, o seu pressuposto, como a sua prioridade neste caso escolheram dois tipos que são:

✓ Essencial – é desejável fazer com que as funcionalidades já existentes funcionem corretamente.

✓ Obrigatório – é obrigatório ser feito, foi pedido no projeto.

Nome do Caso de uso	Fazer Login
Descrição	Este caso de uso consiste na introdução, por parte de um utilizador, num conjunto de nome de utilizador / palavra-chave, que permita ser reconhecido no sistema e obter as suas atuais permissões no mesmo.
Ator	Utilizador
Prioridade	Essencial
Pressuposto	O servidor encontra-se ativo e o programa em execução.
Pré-condição	O conjunto nome de utilizador/palavra-chave tem de ser válido. A ligação à base de dados tem que estar disponível.
Pós-condição	O utilizador tem acesso ao sistema de acordo com as permissões registadas no sistema.

Tabela 6 Fazer Login

Nome do Caso de uso	Inserir informação de documentos
Descrição	Este caso de uso consiste na introdução dos dados de informação dos documentos
Ator	Utilizador
Prioridade	Essencial
Pressuposto	O servidor encontra-se ativo com o programa servidor em execução.
Pré-condição	O conjunto de informação relativo ao documento é validado e não se sobrepõe a nenhum dos documentos já existente. A ligação à base de dados encontra-se disponível.
Pós-condição	

Tabela 7 Inserir informação de documentos

Nome do Caso de uso	Modificar informação de documentos
Descrição	Este caso de uso consiste na modificação dos dados de informação dos documentos
Ator	Utilizador
Prioridade	Essencial
Pressuposto	O servidor encontra-se ativo com o programa servidor em execução.
Pré-condição	O conjunto de informação relativo ao documento é validado e sobrepõe ao documento que está a ser modificado. A ligação à base de dados encontra-se disponível.
Pós-condição	

Tabela 8 Modificar informação de documentos

Nome do Caso de uso	Eliminar documentos
Descrição	Este caso de uso consiste na eliminação dos dados de informação dos documentos.
Ator	Utilizador
Prioridade	Essencial
Pressuposto	O servidor encontra-se ativo com o programa servidor em execução.
Pré-condição	O conjunto de informação relativo ao documento é validado e não se sobrepõe a nenhum dos documentos já existentes. A ligação à base de dados encontra-se disponível.
Pós-condição	

Tabela 9 Eliminar informação de documentos

Nome do Caso de uso	Pesquisar
Descrição	Este caso de uso consiste na pesquisa de informações de documentos
Ator	Utilizador
Prioridade	Obrigatório
Pressuposto	O servidor encontra-se ativo com o programa servidor em execução.
Pré-condição	Existem documentos no sistema. A ligação a base de dados encontra-se disponível.
Pós-condição	

Tabela 10 Pesquisa de documentos

Nome do Caso de uso	Ver documentos
Descrição	Este caso de uso consiste em ver o conteúdo do documento pesquisado.
Ator	Utilizador
Prioridade	Obrigatório
Pressuposto	O servidor encontra-se ativo com o programa servidor em execução.
Pré-condição	Existir documentos no sistema. A ligação a base de dados encontra-se disponível.
Pós-condição	

Tabela 11 Ver documentos

Nome do Caso de uso	Download documentos
Descrição	Este caso de uso consiste fazer o download do documento pesquisado
Ator	Utilizador
Prioridade	Obrigatório
Pressuposto	O servidor encontra-se ativo com o programa servidor em execução.
Pré-condição	Existir documentos no sistema. A ligação a base de dados encontra-se disponível.
Pós-condição	

Tabela 12 Download documentos

A ilustração seguinte ilustra os processos gerais que o administrador dá para administrar o iPortalDoc. Essa parte já esta desenvolvida por parte da empresa.

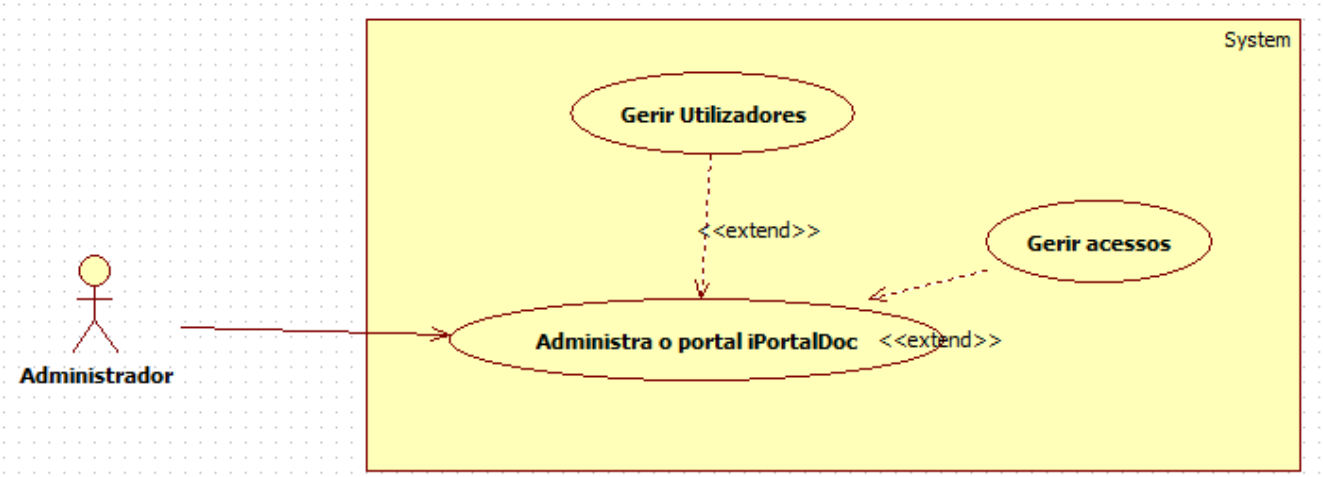

Ilustração 14 Caso de Uso – Administrador

Nome do Caso de uso	Administra o portal iPortalDoc
Descrição	Este caso de uso centra-se na administração do sistema – utilizadores, e acessos
Ator	Administrador
Prioridade	Essencial
Pressuposto	O servidor encontra-se ativo e o programa em execução
Pré-condição	A base de dados está disponível
Pós-condição	

Tabela 13 Administra o portal iPortalDoc

3.1.2. Diagrama de sequência

A ilustração seguinte ilustra a sequência de operações relativas a pesquisar um documento.

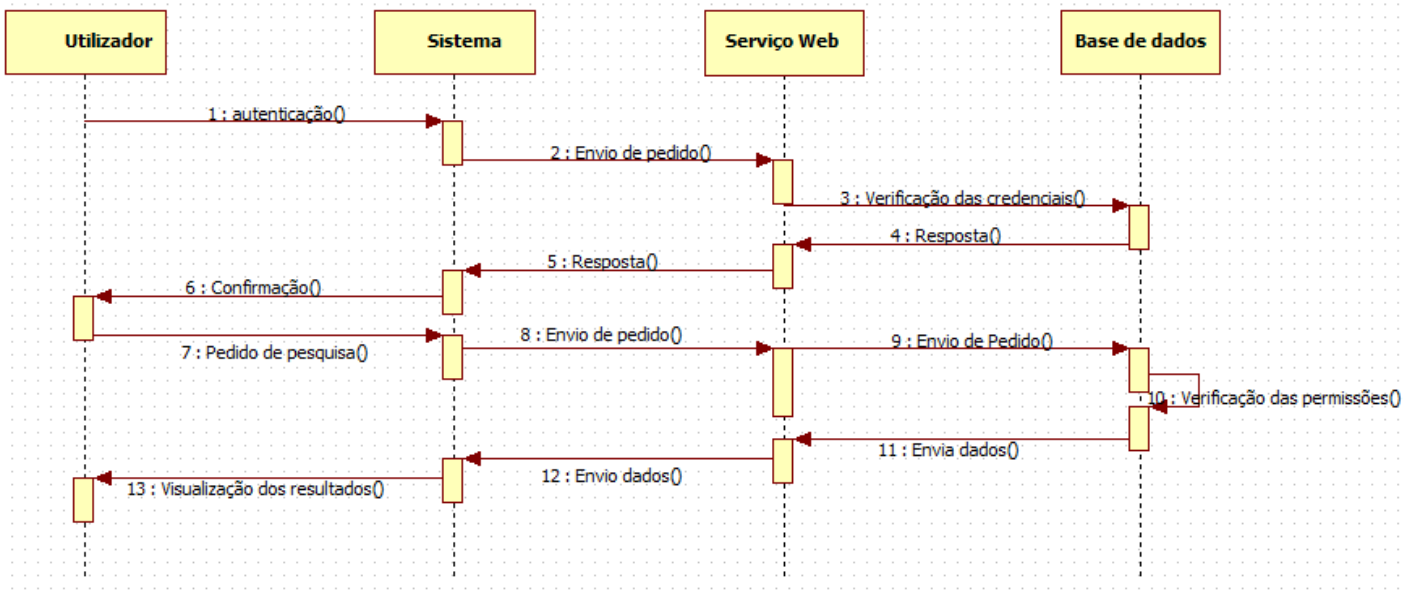

Ilustração 15 Diagrama de Sequencia - Pesquisa de documentos

Inicialmente o utilizador fica autenticado no sistema, e este compara as credenciais com a base de dados através do serviço web. Após as credenciais serem comparadas, o sistema recebe a resposta do serviço e transmite-a ao utilizador. Se o utilizador obtiver resposta positiva, este deve escrever a palavra que pretende pesquisar e depois selecionar a opção "PA" (Pesquisa Avançada). Ao selecionar a opção pesquisa (PA), o sistema envia o pedido ao serviço web e este envia-o para a base de dados. Primeiro vai verificar as permissões do utilizador, caso o utilizador tenha permissão, de ver o documento que deseja, envia os dados para o serviço, e este envio-os para o sistema. Por último o sistema disponibiliza, ao utilizador todos os resultados de uma forma organizada.

3.1.3. Diagrama de Atividade

O seguinte diagrama de atividades pretende exemplificar a sequência de ações ocorridas na pesquisa de documentos.

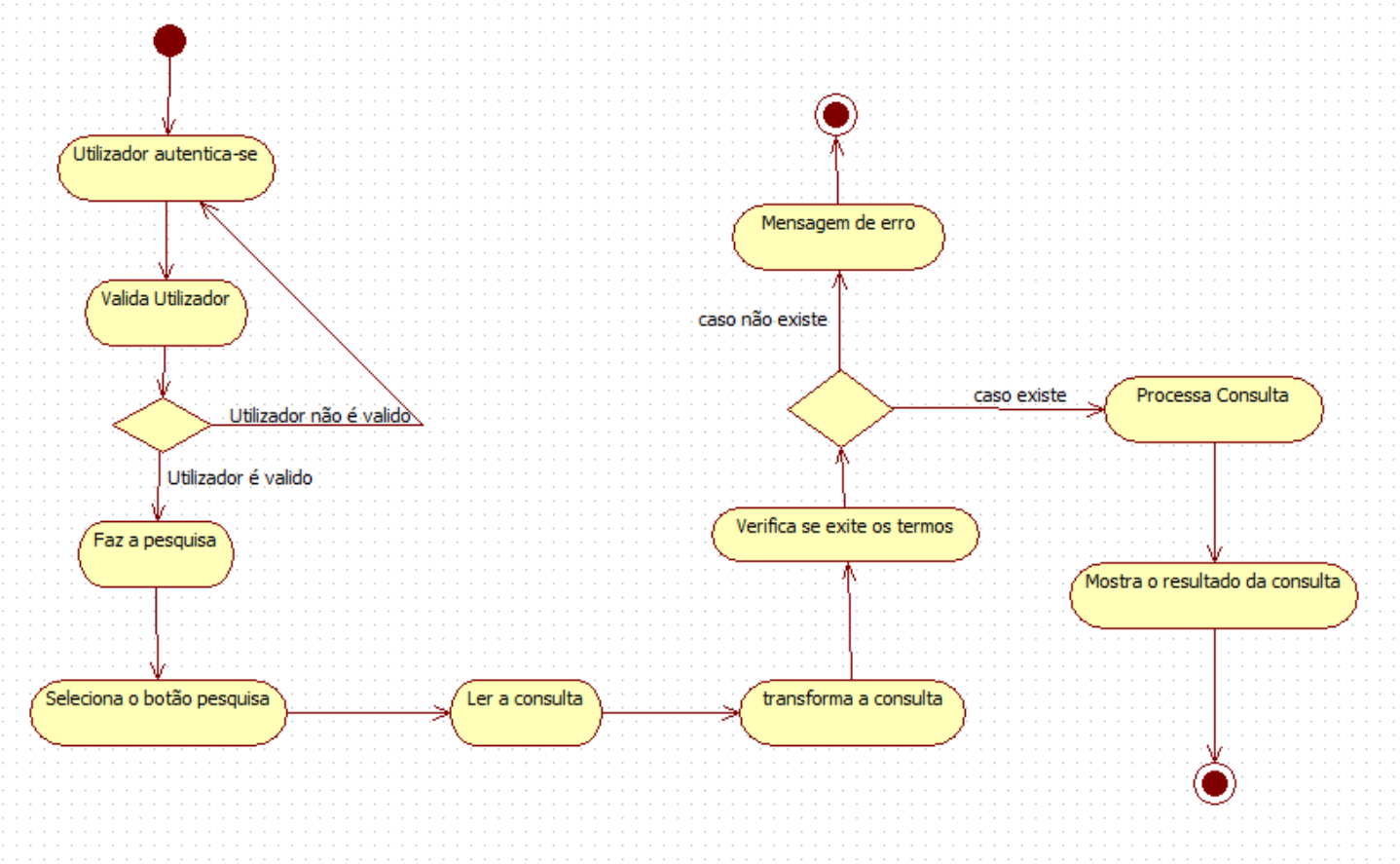

Ilustração 16 Diagrama de atividade - pesquisa

O diagrama de atividade mostra, o fluxo da pesquisa que o utilizador faz. Inicialmente o utilizador autentica-se no portal, caso o utilizador seja, valido mostra a página principal e pode fazer a sua pesquisa. Coloca a frase/termo que o utilizador deseja procurar, e seleciona o botão de pesquisa. Depois o próprio programa é que faz o resto. Primeiro lê a consulta, que o utilizador introduziu, transforma a consulta em *sql* e verifica se existe a frase/termo, e vai trazer as informações relativas à frase/termo. Processa a consulta, e mostra o resultado final da pesquisa ao utilizador. Caso não exista a frase/termo, que esta está a pesquisar aparecerá uma mensagem de erro.

3.2. Requisitos Não funcionais

- Usabilidade
  - O sistema deverá ser simples, intuitivo e a sua informação deve ser de fácil modificação.
  - Não deve exigir conhecimentos informáticos, específicos pelo lado do utilizador comum.
  - Utilizar linguagem e standards aplicados na empresa relativo a software:
    - Sistema Operativo: Linux
    - Base de Dados: Postgres e Oracle
    - Programação: PHP, CSS, JavaScript, AJAX
- Escalabilidade
  - O sistema deverá ser concebido de maneira, a permitir uma fácil manipulação, e alteração das funcionalidades. Poder fazer a integração da ferramenta com o projeto existente iPortalDoc. Este requisito terá maior relevância para o cliente, que poderá assim numa fase posterior modificar ou aumentar o produto mediante as suas necessidades.
- Fiabilidade
  - O sistema deverá ter boa fiabilidade, isto é, excluindo problemas relacionados, por exemplo, com o servidor onde o sistema está alojado, deverá estar sempre disponível.
- Desempenho
  - Efetuar a operação pesquisa de forma a assegurar que as comunicações efetuadas não sofram atrasos inadequados, ou seja, não demorar mais que 5 minutos a dar os resultados da pesquisa pretendida.
- Segurança
  - Deverá garantir a proteção dos dados transmitidos e armazenados;
  - Assegurar que não seja permitido o acesso não autorizado ao sistema e aos dados;

As medidas de segurança são implementadas com as permissões que cada utilizador tem acesso, todos têm acesso ao iPortalDoc mas não têm todos as mesmas permissões para a visualização. As permissões dos utilizadores podem ser Leitor, SuperUtilizador, Administrador, Coordenador. Se o utilizador for Leitor, e o documento tiver na pasta que tem a permissão SuperUtilizador esse utilizador não pode visualizar esse documento.

Todos esses requisitos não funcionais foram aplicados e testados no projeto.

3.3 Arquitetura do Sistema

A arquitetura física do sistema implementado é baseado no modelo cliente-servidor, tendo como interface de comunicação um browser vulgar acedido via internet. Na ilustração abaixo descreve-se de forma mais detalhada a sua composição.

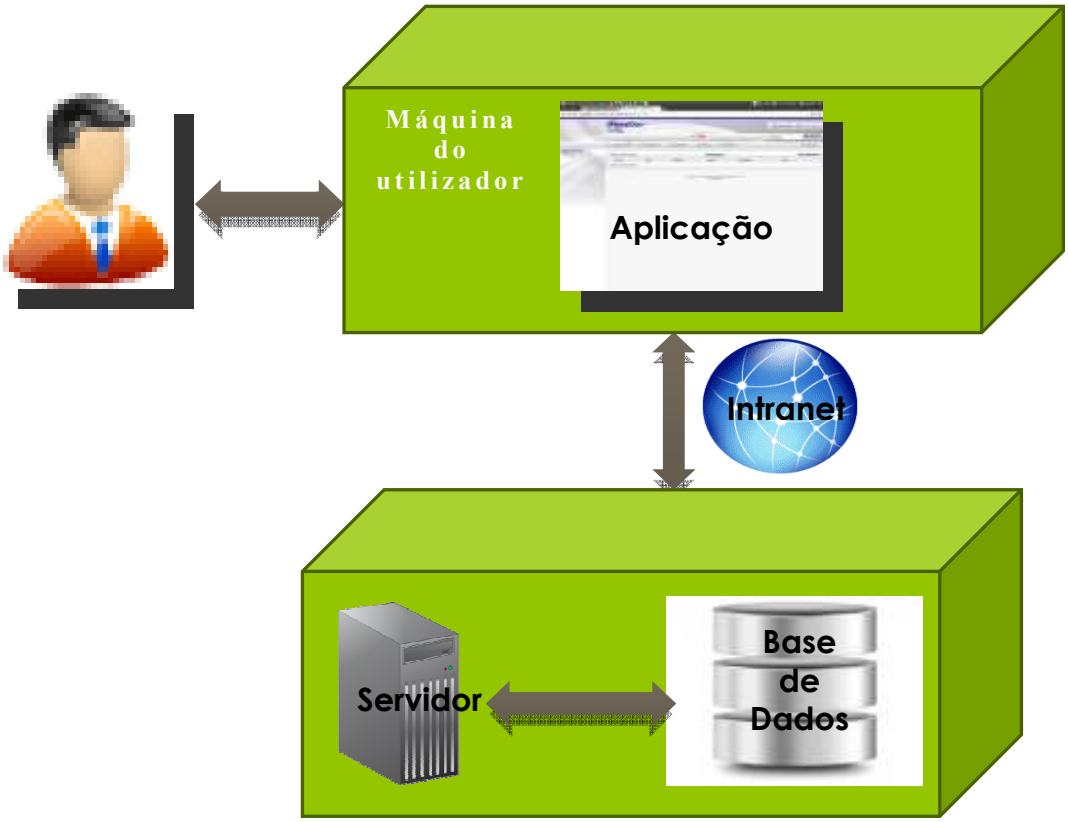

Ilustração 17 Arquitetura do Sistema

Descrição dos componentes:

- Máquina do utilizador: O utilizador, depois de autenticar acede pela sua máquina ao portal. Necessita de ter um browser com suporte de CSS, JavaScript, e ajax. O acesso será feito via rede de comunicação interna.
- Servidor: No servidor reside a aplicação que recorre a uma base de dados – Postgres ou Oracle. O servidor necessita de executar instruções em javaScript e *query* de *sql* e suportar CSS.

A Informação que é trocada entre a aplicação e a base de dados são os dados guardados e os dados que o utilizador armazena na base de dados, para depois fazer a pesquisa. As restrições que existem, são todos os utilizadores de uma determinada empresa, que têm de ter as suas credenciais e as suas permissões. Se tiverem permissões de escrita, leitura e eliminação, o

utilizador pode inserir, modificar e eliminar os dados. Mesmo que tenham essas permissões podem não ter acesso a alguns documentos por causa das permissões que puseram a esse documento.

Ao fazer a pesquisa a *query* que é mandada para o servidor e para a base de dados tem em primeiro lugar que fazer uma verificação para saber se o utilizador em causa tem permissão de visualizar e fazer downlaod de algum documento.

4. Descrição do projeto

4.1. Estrutura da base de dados

Foi utilizada a BD já existente no sistema, onde foram feitas alterações necessárias para o projeto de pesquisa avançada. O diagrama em baixo apresentado, só mostra a parte das tabelas que é preciso para a pesquisa.

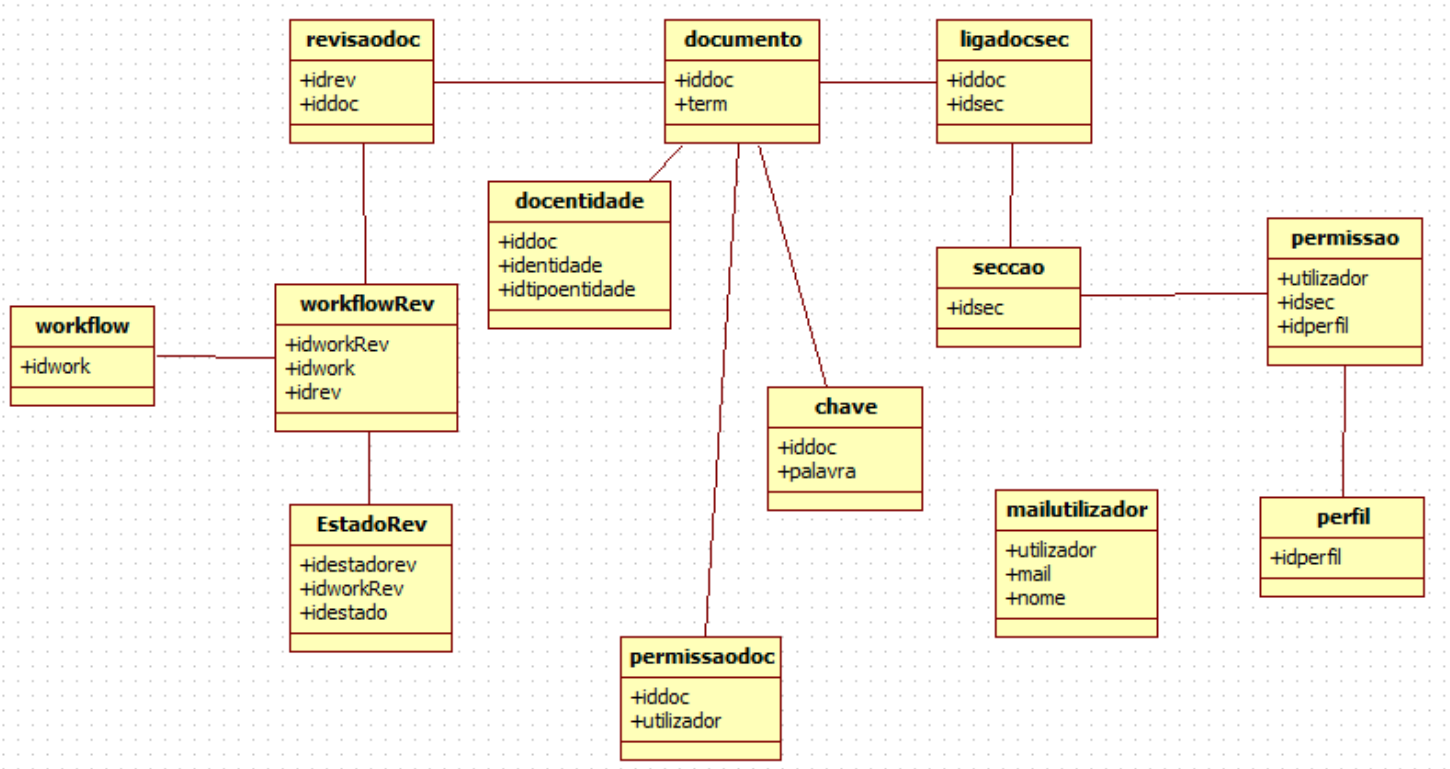

Ilustração 18 Diagrama de ER - base de dados

Passo a explicar cada componente da tabela:

Documento, trata de armazenar todas as informações relativas ao documento. A tabela *chave* é armazenada às palavras-chave relativas ao documento, quando o utilizador insere. A tabela *permissaodoc* é onde esta guardado o que tem de permissão o documento. A tabela *docentidade* é onde está guardada a entidade que está associada ao documento. Todos os documentos estão associados a uma revisão que é guardada na tabela *revisaodoc*, quando o utilizador faz uma alteração as informações de um documento, é criada uma nova revisão, mas na tabela documento não é criada um novo documento. A tabela *revisaodoc* está associada a tabela *workflowrev* que esta associada ao *workflow*. Todos os documentos estão associados a um workflow que é guardado na tabela *workflow*. À tabela e*stadoRev* está armazenado o estado da revisão. Um documento está associado a uma seção que é armazenada à informação das tabelas

ligasec e *secção*. Cada seção está associada às permissões e ao perfil que esta guarda nas tabelas *permissão*, que diz quais são as permissões da seção e quais dos utilizadores que tem acesso. A tabela *perfil* diz qual é o perfil da seção. Por último a tabela *mailutilizador* guarda todas as informações do utilizador.

Para acrescentar o campo *termo* do tipo *tsvector*, na tabela *documento* é preciso fazer algumas configurações na base de dados, para poder utilizar este tipo tsvector, irei mostrar os passos que é preciso fazer.

4.2. Configuração da base de dados

Para instalação ou alteração da base de dados, necessária ao funcionamento da solução apresentada no diagrama E-R, é disponibilizado juntamente com a aplicação um script de SQL, que permite as alterações e mecanismos necessários, para a correta utilização. Os sistemas de gestão de base de dados, permitem importar scripts, assim sendo, dependendo do utilizador é necessário proceder, à sua importação de forma que este disponibiliza. Após esta importação é necessário configurar o projeto, que utilizará a aplicação, tendo acesso à BD em questão, ou seja, fornecer o provedor e configurar acessos, algo que depende da SBDB utilizada, que já está criado no projeto anteriormente.

O projeto vem inicialmente capaz de aceder à base de dados Postgres, como a biblioteca Tsearch2 e a base de dados Oracle.

4.2.1. Instalação da biblioteca Tsearch2

A biblioteca *Tsearch2* já vem quando instalamos o Postgres, o que não está feito é a instalação dos tipos, os índex próprios dessa biblioteca. O *Tsearch2* é basicamente uma biblioteca de funções em C e compiladas como Shared Object (.so). A biblioteca está localizada na pasta ***/usr/share/postgres/8.1/contrib***, no meu caso, pode estar localizada noutra pasta. As funções da biblioteca do Tsearch2 estão disponíveis para o uso no postgres. No entanto, ainda é necessário que essas funções sejam registadas no banco de dados onde pretendemos usar as funcionalidades.

O *Tsearch2* é composto por alguns componentes, personalizados comuns de base de dados, como tabelas, tipos e operadores.

Para registar as funções na base de dados e criar os demais objetos, o *Tsearch2* coloca um arquivo com extensão ***.sql*** dentro da pasta ***/usr/share/postgres/8.1/contrib*** da instalação do

PostgreSQL.

Para instalar o Tsearch2 na base de dados *iportaldoc*, que já existia anteriormente, usaremos o arquivo ***.sql*** fornecido pelo *Tsearch2* para registar as funções e criar todos os objetos necessários para o funcionamento da biblioteca.

Para fazer todos os passos usaremos a linha de comandos da seguinte maneira:

```
psql $_dbname – f /usr/share/postgres/8.1/contrib/tsearch2.sql
```

$_dbname – nome da base de dados.

Com a instalação do *tsearch2*, na base de dados que estamos a utilizar, já podemos fazer as alterações necessárias.

4.2.2. Postgres

A alteração que foi feita na base de dados foi adicionar a coluna *terms* do tipo *tsvetor*, e criar o *índex* na coluna adicionada do tipo *gist*.

O código necessário para adicionar, a coluna e criar o *index* é o seguinte:

```
ALTER TABLE documento ADD COLUMN terms tsvector;
CREATE INDEX documento_terms_idx ON documento USING gist(terms);
```

Depois de feitas estas alterações temos de fazer um *update* da coluna da tabela documento, de seguida vamos buscar as colunas que queremos guardar em termos, na nova coluna *terms,* como mostra o código abaixo.

```
update documento set terms = to_tsvector
(dropatsymbol(simples(coalesce(titulo_pesquisa, '') || ' '||
coalesce(descricao_pesquisa,'') || ' ' || coalesce(sumario_pesquisa,'') ||' '||
coalesce(chaves_pesquisa, '');
```

Foi criado um *trigger* para quando fazemos um *insert* ou um *update* aos documentos, este guarda os termos na coluna automaticamente

4.2.3. Oracle

Já para a base de dados Oracle é utilizado o Oracle Text.

O Oracle Text é uma ferramenta, que permite a criação de aplicativos, de texto de consulta,

e classificação de documentos. O oracle text fornece palavras, indexação e pesquisa por termos, e capacidade de visualização para o texto.

Uma regra da empresa, é que a base de dados tem que ser igual: o que na base de dados Postgres tem, a base de dados Oracle também tem, mesmo que o campo não seja utilizado, como acontece aqui. Foi criado o campo *terms* do tipo *varchar*, e foi criado os índex nos campos *título*, *sumario_pesquisa*, *descrição_pesquisa*, *chave_pesquisa* do tipo *CTXSYS.CONTEXT,* da tabela *documento*.

```
ALTER TABLE documento ADD terms varchar(20);

CREATE INDEX documento_titulo_idx ON documento(titulo_pesquisa)
INDEXTYPE IS  CTXSYS.CONTEXT
PARAMETERS('sync (on commit)');
```

4.2.4. Script

Foi criado um script em php, para fazermos as alteração à base de dados necessárias, como foi explicado em acima, para termos um funcionamento correto na aplicação. Esse script está feito para a base de dados Postgres e para Oracle, mas se for necessário acrescentar outro tipo de base de dados pode ser feito. Esse script só é executado uma vez quando fazemos uma instalação ou uma atualização, isto quando o utilizador o programa.

O comando para executar esse script na linha comando é:

```
php 0018-20120803-search_engine.php /etc/iportaldoc/default.cfg
```

4.3. Programação

Já existia código do programa iPortaldoc feito. O código estava dividido, tinha as liberaria de fazer a comunicação com a base de dados, classes para cada objeto, por exemplo, tinha uma classe chamada *documento.php*, e por ai adiante. O que tinha de fazer era um motor de pesquisa dentro do iPortalDoc que já estava feito.

Já haviam sido criados diretórios por parte da empresa e foram utilizados para o projeto que desenvolvi. Irei apresentar todas as classes criadas de raiz e as classes que foram acrescentados ao código. Antes de dizer o que faz cada classe irei apresentar a estrutura dos ficheiros que fazem parte do projeto.

- work/dbdoc
- include
    - top
        - pesquisa_chave-php
        - top.php
        - funcoes_pesq_termo.php
        - pesquisa_termo.php
        - pesquisa_sem_termo.php
        - result_pesquisa_sem_termo.php
        - result_pesquisa_termo.php
- PHP
    - IfDBDocumento.phpclass
- LIB
    - libdatabase.php
    - liboracle.php
    - libpostgres.php
- site
    - js
        - pesquisa
            - GeneralSearch.js
    - traducao
        - alemao.txt
        - arabic.txt
        - english.txt
        - espanhol.txt
        - frances.txt
        - italian.txt
- SQL
    - 0011_versao_4_0
        - 0018-20120803-search_engine.php

Na diretoria include

Na diretoria *include* apresenta-se diversas pastas que contêm diversos ficheiros de php, a pasta que foi acrescentado e alterados ficheiros foi na pasta *top*.

Na sub-diretoria top

Nesta subdiretoria apresenta-se todos os ficheiros que fazem parte da pesquisa, para mostrar e para chamar os ficheiros necessários para o funcionamento da pesquisa termo.

top.php

Este ficheiro consiste na visualização do botão na página principal, escrever o termo e carregar no botão pesquisa (PR) abre uma nova janela e mostra o resultado.

Código necessário para mostrar o botão e o caminho para abrir uma nova janela e mostrar o resultado.

```
<a id="btPag" name="btPag"
href="javascript:reloadPagina(document.pesquisaRapida);"
title="<?=translate_lang("Pesquisa")?>" >
        <div id="quick_find_but"><?=translate_lang("P")?></div>
</a>
```

Pesquisa_chave

Este ficheiro é chamado quando o botão, é carregado para fazer a pesquisa. Neste ficheiro verifica-se qual é o tipo de pesquisa que o utilizador escolheu, se foi pesquisa rápida, ou pesquisa avançada (essas duas pesquisas já existem) e a pesquisa por termos (desenvolvida no estágio).

funções_pesq_termo

Verifica-se que o utilizador escreveu alguma palavra ou não. Caso o utilizador digitalizou um termo vai chamar o ficheiro *pesquisa_termo* e depois chama o ficheiro *result_pesquisa_termo* para mostrar o resultado da pesquisa. Caso o utilizador não escreveu algum termo vai chamar o ficheiro *pesquisa_sem_termo* e depois chama o ficheiro *result_pesquisa_sem_termo*.

pesquisa_termo

Este ficheiro é chamado se o utilizador escreveu alguma palavra. Este ficheiro vai chamar a função *pesquisa_termo_avançado* no ficheiro *PHP/IFDBDocumento.phpclass* para fazer a pesquisa na base de dados. Também é este ficheiro que contém o código para a palavra digitada aparecer a vermelho.

result_pesquisa_termo

Este ficheiro mostra o resultado final da pesquisa.

pesquisa_sem_termo

Este ficheiro só é utilizado quando o utilizador não escreve nenhuma palavra. O funcionamento é o mesmo que o ficheiro *pesquisa_termo.php*.

result_pesquisa_sem_termo

Este ficheiro só é utilizado quando o utilizador não escreve nenhuma palavra. O funcionamento é o mesmo que o ficheiro *result_termo.php*.

Na diretoria PHP

Nesta diretoria foi alterado a classe *IfDBDocumento*, foi acrescentada a função de pesquisa de *Full Text Search*.

O código desta função é o mesmo da pesquisa avançada com algumas alterações para a utilização do método *Full Text Search*.

Em primeiro lugar vai verificar se o utilizador tem permissão de ver o documento que conte a palavra digitada. Depois é que vai buscar a última revisão dos documentos que o utilizador tem acesso e que contém a palavra digitada.

A seguinte linha vai fazer a pesquisa na coluna *terms* com a palavra que foi digitalizada.

```
$filter_term = $GLOBALS["libdatabase"]->tsquery($pal);
```

A linha abaixo faz o ranking da palavra a pesquisar:

```
$rank = ", ".$GLOBALS["libdatabase"]->PageRank($filter_term);
```

A linha de código é responsável por normalizar os termos de consultas, transformando-as no tipo *tsquery()*.

```
$join_tsvector ="AND (".$GLOBALS["libdatabase"]->FullTextSearch($filter_term).")";
```

Essas linhas foram juntas a query para a pesquisa ser mais rápida e mais eficiente. Ao mostrar o resultado ao utilizador mostra por ordem descendente.

Na diretoria LIB

libdatabase

Nesta classe são criados todas as funções para fazerem a ligação a base de dados, é por esta classe vão ser chamadas as outras classes como liboracle e libpostgres. Assim o código pode funcionar para qualquer base de dados.

Esta classe foram acrescentads três funções que são:

- ✓ FullTextSearch: esta função faz a conversão do *full texto searcher* entre a base de dados. Neste caso na base de dados Postgres utilizamos o Tsearch e no Oracle é o Oracle Text.
- ✓ Tsquery: para fazer a pesquisa na coluna terms a palavra digitalizada pelo utilizador.
- ✓ pageRank: esta função dá-nos o ranking, dependo da base de dados.

O código a seguir apresentado é do FullTextSearch:

```
function FullTextSearch($palavra) {
        switch ( $this->databaseType){
                case "POSTGRES":
                   return $GLOBALS["libpostgres"]->pg_FullTextSearch($palavra);
                   break;
                case "ORACLE":
                   return $GLOBALS["liboracle"]->oci_FullTextSearch($palavra);
                        break;
                /*base de dados por defeito*/
                default:
                   return $GLOBALS["libpostgres"]->pg_FullTextSearch($palavra);
                break;
        }
  }
```

Esta parte do código é para mostrar como podemos ter vários tipos de base de dados a funcionar corretamente. Aqui é apresentado a base de dados Postgres "libpostgres" que chama o ficheiro que tem todas as funções, por exemplo ligar a base de dados, o insert, etc. Mas primeiro cria-se uma função genérica em libdatabase e depois vamos às classes das bases de dados e fazemos o código correspondente a essa base de dados. Mas há uma regra importante, o que deveconter uma base de dados e a outra base de dados.

Os ficheiros liboracle.php e libpostgres.php são chamados dependo da base de dados que é utilizada pelos clientes. Contém as mesmas funções que o ficheiro libdatabase.php só que são utilizados as funções correspondentes a cada base de dados.

libpostgres

pg_FullTextSearch a sintaxe é *terms@@$palavra*. A pesquisa é feita na coluna terms da tabela document.

```
Select * From documento where terms @@ to_tsquery ('postgresql')
```

O operador like é o @@ no tsearch, para verificar a ocorrência da palavra-chave no campo indexado *terms*.

pg_tsquery a sintaxe

```
to_tsquery('".$palavra."')
```

Esta classe a responsável por normalizar os termos de consultas, transformando-as no tipo – tsquery(). Baseia-se na pesquisa de correspondências existentes entre o objeto de consulta, constante em uma função de consulta – tsquery(), e os dados disponibilizados por uma função de representação de dados do tipo tsvetor.

pg_PageRank: a sintaxe é *rank (terms, $palavra) as rank*. Esta função gera um número que representa a classificação do registo na nossa consulta.

```
Select titulo, rank($terms, to_tsquery('ruby')) as rank
From document
Where vectors @@ to_tsquery ('ruby')
```

liboracle

oci_FullTextSearch a sintaxe é :

```
contains(titulo_pesquisa, '".$palavra."', 1)>0
OR contains(sumario_pesquisa, '". $palavra."', 2)>0
OR contains(descricao_pesquisa, '". $palavra."', 3)>0
OR contains(chaves_pesquisa, '".$palavra."', 4)>0
```

A pesquisa é feita nas colunas titulo_pesquisa, sumario_pesquisa, descrição_pesquisa, chaves_pesquisa da tabela documento.

oci_tsquery a sintaxe

```
$pal = str_replace ("&" , "and", $palavra);
$search = "$pal";
```

Nesta função não precisamos usar o tsquery porque na base de dados oracle não existe. Mas os operadores booleanos são diferentes em Postgres para Oracle, por isso nessa função em Oracle fazemos a conversão dos operadores booleanos ***&*** para ***and*** e | para ***or***.

oci_PageRank: a sintaxe é:

```
$rank = "(Score(1) + Score(2) + Score(3) + Score(4)) as rank"
```

Esta função gera um número que representa a classificação do registo na nossa consulta.

4.4. Operadores booleanos

Existe diferenças na representação dos operadores booleanos no postgresql e no oracle. Como podemos ver na seguinte tabela.

Operador	Postgresql	Oracle
AND	&	and
OR	|	or

Tabela 14 Tabela de operadores booleanos

Diretoria site/js

Nesta sub-diretoria site/js foi criada a pasta de *pesquisa*, e dentro dessa pasta foi criado o ficheiro *GeneralSearch*.

O princípio deste ficheiro é fazer o loading quando estamos à espera dos resultados da pesquisa e para abrir a janela de pesquisa. Neste ficheiro encontram-se várias funções:

A função *Loading()* chama outra função *showLoading()* do ficheiro *libGenericLayout* para mostrar o loading. Esta função não funciona sozinha, só funciona quando é chamada pela outra função.

A função *reloadPesquisa()* é utilizada quando clicamos no botão ok, esta função tem como princípio enviar para o servidor, o pedido dos resultados da pesquisa, e quando espera pelos resultados mostra o loading ao chamar a função Loading().

A função *errorRequest()* só é ativada quando houver erros.

A função *reloadPagina()* é utilizada no botão P (Pesquisa por termos) na página principal. Esta função abre uma nova janela para mostrar os resultados da palavra digitada.

A função *reloadBody()* é utilizada para fazer loading até mostrar os resultados da palavra digitada.

A função *clos_wainting_panel()* é utilizada para esconder o loading. Esta função é ativada quando os resultados aparecerem ao utilizador.

Na diretoria site/traduções

Os ficheiros *alemao.txt*, *arabic.txt*, *english.txt*, *espanhol.txt*, *frances.txt*, *italina.txt*, portugues.txt foram acrescentados ao texto que é preciso para fazer a tradução para as diferentes línguas utilizadas no programa.

5. Cronograma

O cronograma seguinte ilustra o esforço estimado para cada uma das fases do projeto, bem como a sua sequência:

Nome da Tarefa	Duração	Início	Conclusão
1 Motor de Busca	**241 dias**	**Seg 09-04-12**	**Sáb 09-03-13**
1.1 Estudo do SGDW	**9,88 dias**	**Seg 09-04-12**	**Sex 20-04-12**
1.1.1 Leituras dos manuais sobre a utilização do iPortalDoc	2 dias	Seg 09-04-12	Ter 10-04-12
1.1.2 Funcionamento do programa	8 dias	Qua 11-04-12	Sex 20-04-12
1.1.3 Relatorio sobre o funcionamento do iPortalDoc	7 dias	Qui 12-04-12	Sex 20-04-12
1.2 Investigação de métodos de pesquisas em base de dados	**35 dias**	**Seg 09-04-12**	**Seg 28-05-12**
1.2.1 Relatorio com as soluções de metodos pesquisa em base de dados	35 dias	Seg 09-04-12	Seg 28-05-12
1.3 Analise do codigo e base de dados	**34,88 dias**	**Seg 07-05-12**	**Sex 22-06-12**
1.3.1 Analisar a estrutura da base de dados	25 dias	Seg 07-05-12	Sex 08-06-12
1.3.2 Analisar o codigo do programa iPortalDoc	12 dias	Qui 07-06-12	Sex 22-06-12
1.4 Projetar o Motor de pesquisa	**5,88 dias**	**Ter 26-06-12**	**Ter 03-07-12**
1.4.1 Desenho grafico	6 dias	Ter 26-06-12	Ter 03-07-12
1.5 Desenvolvimento do motor de pesquisa avançada	**75,88 dias**	**Qua 27-06-12**	**Qua 10-10-12**
1.5.1 Desenvolvimento do codigo	76 dias	Qua 27-06-12	Qua 10-10-12
1.6 Testes ao programa	**16 dias**	**Qua 10-10-12**	**Qua 31-10-12**
1.6.1 Relatorio de Teste	7 dias	Seg 01-10-12	Qua 10-10-12
1.7 Documentação	**230 dias**	**Sáb 21-04-12**	**Sáb 09-03-13**
1.7.1 Site	64 dias	Sáb 21-04-12	Qui 19-07-12
1.7.2 Relatorio inicial	84 dias	Seg 07-05-12	Qui 30-08-12
1.7.3 Poster	20 dias	Ter 18-09-12	Sáb 13-10-12
1.7.4 Artigo	30 dias	Ter 01-01-13	Sáb 09-02-13
1.7.5 Dissertação	91 dias	Seg 05-11-12	Sáb 09-03-13

Ilustração 19 Cronograma

O estudo do SGBD iPortalDoc é importante para esta tarefa porque é aqui que conseguimos perceber como o programa funciona, e o que vai ser preciso fazer com o motor de pesquisa. Esta tarefa envolve a criação de um relatório, do funcionamento do iPortalDoc, onde foi utilizado um exemplo de uma instituição que pretende colocar todos os documentos dos alunos no dito programa. A parte aqui representada foi a de tesouraria, onde podemos ver o relatório (Rocha, Relatório de gestão documental - funcionamento do programa iPortalDoc, 2012).

A tarefa de Investigação de métodos de pesquisas em bases de dados apresenta-se como uma atividade crítica para o desenvolvimento do projeto. A pesquisa bibliográfica relativa a

soluções de motor de pesquisa e de base de dados, é uma tarefa que consome muito tempo mas preciosa, na medida em que nos permitirá identificar boas práticas, eliminar erros e reconhecer pontos importantes relativos à execução de indexação da base de dados e qual será a melhor solução. O tempo despendido nesta tarefa, apesar de longo, trará relevância ao projeto.

Foi realizado um relatório sobre métodos de pesquisa em base de dados e em motores de pesquisa que podemos realizar em (Rocha, Relatório de soluções de pesquisas de palavras-chaves em base de dados, 2012).

A análise do código e base de dados foram tarefas de relevância para o projeto. Na parte do estudo da base de dados é importante saber como está construída a base de dados da aplicação iPortalDoc. Com esta base de dados podemos identificar quais as tabelas que vamos utilizar para a aplicação do nosso projeto. Na parte da análise de código da aplicação do iPortalDoc, é precioso saber como está a ser feito e como liga à base de dados.

O projeto do motor de pesquisa avançado, foi feito pela interface gráfica necessáriao para a aplicação.

Na tarefa de desenvolvimento do código do motor de pesquisa foi implementada a base de dados. Nesta tarefa houve um faseamento porque a meio do projeto foi mudada a técnica que estava a ser utilizada para outra técnica, por isso tive que estudar como implementar a nova técnica.

Na tarefa testes ao programa foi elaborado um relatório com os testes necessários para saber como o programa funcionava e o que era pretendido. Essa tarefa não foi pensada, antes foi falada já quase no fim do programa, que era uma tarefa precisa.

A tarefa documentação, que realiza todos os relatórios para a faculdade, também foi feita por faseamento nas entregas, uma vez que precisava de alguns aspetos ser feito ao pormenor, e demorou algum tempo.

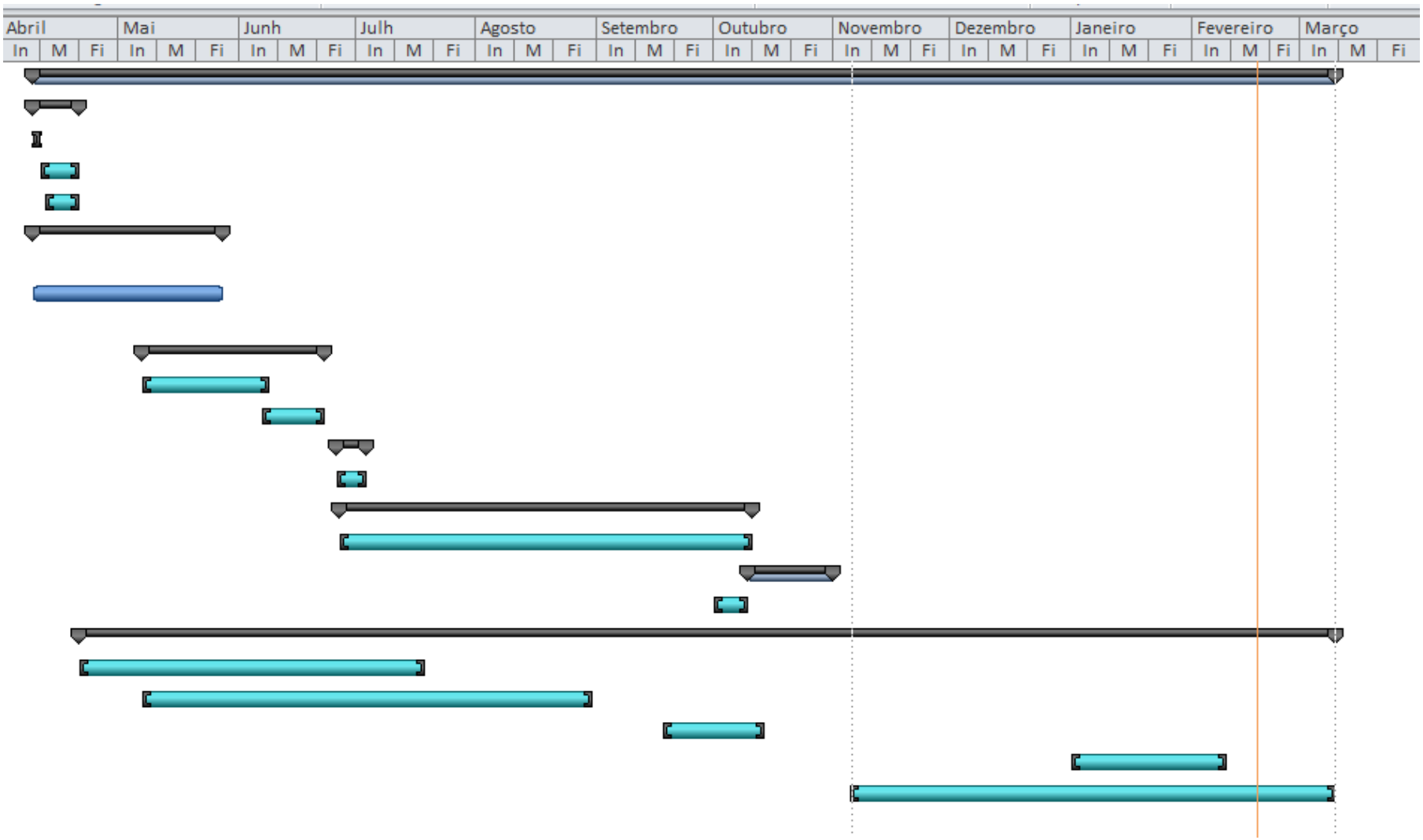

Ilustração 20 Modelo de Grantt

6. Meios previstos e meios necessários

Os meios necessários à realização deste projeto podem ser divididos entre recursos técnicos, recursos de equipamento e recursos humanos.

Os recursos tecnológicos prendem-se com as necessidades, relacionadas com a tecnologia a implementar.

- ✓ SGBD requisitado:
  - Postgred;
  - Oracle
- ✓ Linguagem de programação:
  - HTML;
  - PHP;
  - AJAX
  - JavaScript;
- ✓ Sistema operativo:
  - Ubuntu.

Uma vez que se trata de software livre, não existem custos associados à utilização destas tecnologias, a licença do software é GNU (General Public License).

Os recursos de equipamento são de forma sucinta os seguintes:

- ✓ Servidores que contêm a base de dados;
- ✓ Infra-estrutura de acesso ao servidor;
- ✓ Computar para o acesso ao servidor.

Para o desenvolvimento do projeto, a empresa em que estou a estagiar disponibiliza todas as condições mencionadas em cima. Deveremos considerar uma percentagem de todos os custos associados à manutenção destes equipamentos como custos do projeto.

Os recursos humanos disponibilizados para o projeto foram os seguintes: a aluna que desenvolve o projeto, o orientador interno e a orientadora externa (empresa).

A estes recursos, estão associados os custos relativos ao tempo despendido, no desenvolvimento do projeto, tempo utilizado pelo orientador, na sua função, bem como a orientadora da empresa onde estou a estagiar.

7. Problemas e Decisões

A grande decisão foi durante o estágio quando faltava mais ou menos 2 meses para acabar, foi mudado o método que estava a ser utilizado no projeto.

O primeiro método foi o *document-term matrix* é uma matriz que descreve a frequência dos termos, que ocorrem em uma coleção de documentos. Numa matriz term-document, linhas correspondem aos documentos da coleção, e as colunas correspondem aos termos. Existem vários sistemas para a determinação, do valor que cada entrada na matriz deve ter. Com esse método percebi que quanto mais documentos, tinha na base de dados, mais lento ficava por essa razão, foi mudado o método para *Full Text Search*. Antes de recomeçar fazer o trabalho, com este método foi preciso fazer um estudo para saber como funcionava, em diversas bases de dados, no meu caso na base de dados Postgres e Oracle.

Para utilizar este método foi preciso, fazer alterações na base de dados, e algumas alterações no código. No postgres como foi visto nos capítulos anteriores, foi necessário utilizador a libaria Tsearch2, já no Oracle não foi preciso configurar nada.

8. Análise de resultados

8.1 Plano de teste

As funcionalidades a testar são:

Identificador	Caso de Teste	Descrição
CT1	Pesquisar por termos	Pretende-se que mostre as informações dos documentos correspondentes.
CT2	Pesquisar por frases	Pretende-se que mostre as informações dos documentos correspondentes
CT3	Pesquisar por operadores booleanos (& and e | or)	Pretende-se que mostre as informações dos documentos correspondentes
CT4	Pesquisar por termos de casos especiais	Pretende-se que mostre as informações dos documentos correspondentes
CT5	Escrita de informação do documento	O utilizador deve poder escrever na base de dados.
CT6	Atualizar a informação do documento	O utilizador deve poder atualizar na base de dados.

Tabela 15 Funcionalidades a testar

8.2. Funcionalidades a não testar

As funcionalidades a não testar são os módulos:

- ✓ Workflow;
- ✓ Diretoria;
- ✓ Definições;
- ✓ Documento
  - Encaminhar;
  - Check out;
  - Associar;

- Liga Doc.;
- Mover Docs;
- Cancelar Doc;
- Mails Ass.;
- Workflow Doc;
- Acções Docs.;

As funcionalidades a não ser testadas apresentadas em cima são porque já foram testadas pelos colegas da empresa, e já que não faziam interferência ao meu projeto não havia necessidade de testar novamente destas funcionalidades.

8.3. Abordagem

Pretende-se com este processo estabelecer um relacionamento de confiança e segurança, garantido no produto três caraterístico essencial:

- ✓ Fiabilidade;
- ✓ Usabilidade;
- ✓ Segurança;

Nesta seção vai definir-se como abordar os testes ao software, os itens pretendidos e as funcionalidades a testar. Para efetuar os testes ao produto, iremos utilizar apenas testes unitários. Podemos dividir os testes em dois módulos diferentes.

- ✓ Codificação dos módulos do sistema – Teste Unitário;
- ✓ Cobertura dos requisitos – Teste de Integração.
- ✓ Manutenção do produto – Teste de regressão é essencialmente vocacionados para a fase de desenvolvimento.

8.4. Critérios de sucesso/insucesso

Este desenho de teste avalia um conjunto de itens que estão intrinsecamente ligados às funcionalidades disponibilizadas pelo projeto. Estes itens têm como critério de sucesso a verificação

de consistência entre os resultados esperados e os resultados efetivos de saída. Caso esta consistência se verifique, os testes são aprovados.

8.4.1. Classificação dos erros

Para melhor avaliar a qualidade do produto final, estes são submetidos a uma bateria de testes que pretendem encontrar e identificar defeitos, sendo estes comunicados a equipa de desenvolvimento, desencadeando um processo de correção e melhoramento do sistema.

Estes defeitos, quando encontrados são então classificados consoante uma escala de severidade pré-definida, para permitir a equipa do desenvolvimento debruçar-se sobre os problemas mais importantes e estabelecer prioridades, agilizando os processos de eliminação destas incorreções. A escala utilizada é a seguinte:

Nível 1 – O sistema não funciona.

Nível 2 – Embora existem erros, o sistema continua o seu funcionamento, ainda que deficiente.

Nível 3 – Os resultados apresentados são inesperados e inconsistentes.

Nível 4 – Funciona corretamente.

Nível 5 – Proposta de melhoria ou evolução.

8.5. Necessidades de ambiente

Para os testes serem realizados será necessário o terminal de trabalho e desenvolvimento;

8.6. Riscos e contingências

Aqui são definidos os maiores riscos na execução deste plano teste, bem como planos de contingência para os contornar/minimizar.

Risco	*Plano de contingência*
Desenvolvimento das funcionalidades não fica pronto a tempo de ser testadas devidamente.	Redefinição das prioridades dos testes.
Dificuldades na preparação e configuração do ambiente de testes.	Obter o conhecimento e competências necessárias à preparação do ambiente de testes.
Análise dos testes ser mal definidos.	Redefinição da análise dos testes.

Desenho dos testes mal definidos.	Redefinição do desenho dos testes.
São detetados muitos erros ou erros graves durante os testes	Redefinir a prioridade dos testes. Alocar recursos atribuídos aos testes à investigação dos erros encontrados.
Os testes têm erros.	Dividir os testes de forma a reduzir e a minimizar os erros dos mesmos.

Tabela 16 Riscos nos testes

8.7. Casos de teste

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	ipDoc – CT1
Responsabilidade:	Alexandra Rocha
Objetivo:	Pesquisar por termos
Requisitos Testados ipDoc – CT1 Pesquisar por termos	
Especificação de Entradas Pesquisar o termo "manual"	
Especificação de Saídas Mostrar todos os documentos com a informação do respetivo documento que contém o termo pesquisado.	
Outros	
Dependências Deve existir documentos com as informações.	

Tabela 17 Especificação de Caso de Teste ipDoc-CT1

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	ipDoc – CT2
Responsabilidade:	Alexandra Rocha
Objetivo:	Pesquisar por frase
Requisitos Testados ipDoc – CT2 Pesquisar por frase	
Especificação de Entradas Pesquisar a frase "Manual de utilização do iportaldoc"	
Especificação de Saídas Mostrar todos os documentos com a informação do respetivo documento que contém o termo pesquisado.	
Outros	
Dependências Deve existir documentos com as informações.	

Tabela 18 Especificação de Caso de Teste ipDoc-CT2

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	ipDoc – CT3
Responsabilidade:	Alexandra Rocha
Objetivo:	Pesquisar por termos com operadores booleanos & e | (and e or)
Requisitos Testados ipDoc – CT3 Pesquisar por termos com operadores booleanos & e I	
Especificação de Entradas Pesquisar o termo "manual & teste" Pesquisar o termo "manual | marsi"	
Especificação de Saídas Mostrar todos os documentos com a informação do respetivo documento que contém o termo pesquisado.	
Outros	
Dependências Deve existir documentos com as informações	

Tabela 19 Especificação de Caso de Teste ipDoc-CT3

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	ipDoc – CT4
Responsabilidade:	Alexandra Rocha
Objetivo:	Pesquisar por termos com casos especiais
Requisitos Testados ipDoc – CT1 Pesquisar por termos com casos especiais	
Especificação de Entradas Pesquisar o termo "tes's"	
Especificação de Saídas Mostrar todos os documentos com a informação do respetivo documento que contém o termo pesquisado.	
Outros	
Dependências Deve existir documentos com as informações.	

Tabela 20 Especificação de Caso de Teste ipDoc-CT4

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	ipDoc – CT5
Responsabilidade:	Alexandra Rocha
Objetivo:	Escrever na base de dados
Requisitos Testados ipDoc – CT1 Escrever na base de dados	
Especificação de Entradas Inserir informações de um documento que desejamos e guardar na base de dados com o ficheiro.	
Especificação de Saídas Ao ser guardada a informação com o ficheiro (documento), vai aparecer uma mensagem que diz "o documento foi introduzido" e aparece abaixo as informações introduzidas.	
Outros	
Dependências O utilizar tem de ter associado um tipo de documento e um workflow.	

Tabela 21 Especificação de Caso de Teste ipDoc-CT5

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	ipDoc – CT6
Responsabilidade:	Alexandra Rocha
Objetivo:	Atualizar documento na base de dados
Requisitos Testados ipDoc – CT1 Pesquisar por termos	
Especificação de Entradas Escolher o documento que pretende-se atualizar, e atualiza as informações e guarda.	
Especificação de Saídas Ao ser guardada a informação atualizado com o ficheiro (documento), vai aparecer uma mensagem que diz "o documento foi atualizado" e aparece abaixo as informações introduzidas.	
Outros	
Dependências O utilizador tem de ter associado um tipo de documento e um workflow.	

Tabela 22 Especificação de Caso de Teste ipDoc-CT6

8.8 Relatório da Localização dos Componentes de Teste

8.8.1. Componentes de Testes

Os componentes de testes são todos os módulos e classes definidos na seção caso de testes.

8.8.2. Localização dos componentes de testes

Os componentes estão localizados num repositório online, mais concretamente na seguinte localização: https://iportaldoc.ipdocdev205.net/index.php. Este link só pode ser acedido por membros da equipa de desenvolvimento do iPortalDoc, sendo necessário um conjunto nome de utilizador e palavra chave.

8.8.3. Estado dos Componentes

Os diferentes componentes de teste poderão ter três estados distintos:

- ✓ Em desenvolvimento;
- ✓ Pronto e não testado;
- ✓ Pronto e testado.

Apenas são testáveis os componentes que tiverem o estado "Pronto e testado".

8.9. Resultados dos casos de testes

8.9.1. Informação sobre o ambiente de Testes

Os testes foram realizados no computador da empresa com as seguintes caraterísticas: com 1.9GiB de memória, com um processador Intel® Pentium® 4GP 3.00Gz, possuindo ainda um disco de 45.8GiB, no sistema operativo Ubuntu Release 11.04 (natty) Kernel Linux 2.6.38.10 generic.

8.9.2. Registos de Testes

8.9.2.1. Registo de Teste RT1

8.9.2.1.1. Descrição

Este Teste implementa o caso de teste com o identificador ipDoc-CT1.

8.9.2.1.2. Resultados do procedimento de Teste

O procedimento de teste encerrou normalmente, tendo sido escritas as informações dos documentos na base de dados. O teste foi aprovado.

8.9.2.1.3. Evento Anómalos

Não foram registadas anomalias durante a execução de testes.

8.9.2.2. Registo de Teste RT2

8.9.2.2.1. Descrição

Este Teste implementa o caso de teste com o identificador ipDoc-CT2.

8.9.2.2.2. Resultados do procedimento de Teste

O procedimento de teste encerrou normalmente, tendo sido escritas as informações dos documentos na base de dados. O teste foi aprovado.

8.9.2.2.3. Evento Anómalos

Não foram registadas anomalias durante a execução de testes.

8.9.2.3. Registo de Teste RT3

8.9.2.3.1. Descrição

Este Teste implementa o caso de teste com o identificador ipDoc-CT3.

8.9.2.3.2. Resultados do procedimento de Teste

O procedimento de teste encerrou normalmente, tendo sido escritos as informações dos documentos na base de dados. O teste foi aprovado.

8.9.2.3.3. Evento Anómalos

Não foram registadas anomalias durante a execução de testes.

8.9.2.4. Registo de Teste RT4

8.9.2.4.1.Descrição

Este Teste implementa o caso de teste com o identificador ipDoc-CT4.

8.9.2.4.2. Resultados do procedimento de Teste

O procedimento de teste encerrou normalmente, tendo sido escritas as informações dos documentos na base de dados. O teste foi aprovado.

8.9.2.4.3. Evento Anómalos

Não foram registadas anomalias durante a execução de testes.

8.9.2.5. Registo de Teste RT5

8.9.2.5.1. Descrição

Este Teste implementa o caso de teste com o identificador ipDoc-CT5.

8.9.2.5.2. Resultados do procedimento de Teste

O procedimento de teste encerrou normalmente, tendo sido escritas as informações dos documentos na base de dados. O teste foi aprovado.

8.9.2.5.3. Evento Anómalos

Não foram registadas anomalias durante a execução de testes.

8.9.2.6. Registo de Teste RT6

8.9.2.6.1. Descrição

Este Teste implementa o caso de teste com o identificador ipDoc-CT6.

8.9.2.6.2. Resultados do procedimento de Teste

O procedimento de teste encerrou normalmente, tendo sido escritas as informações dos documentos na base de dados. O teste foi aprovado.

8.9.2.6.3. Evento Anómalos

Não foram registadas anomalias durante a execução de testes.

9. Conclusões

Tendo o projeto sido desenvolvido segundo os objetivos propostos, é possível afirmar que os mesmos foram atingidos.

O projeto desenvolvido é um sistema fácil de ser utilizado e os testes realizados focaram-se em avaliar os resultados obtidos através de consultas realizadas. Criou-se um ambiente para serem realizados testes e através de consultas foram verificados os documentos retornados.

Um dos objetivos era que fosse possível usar diferentes tecnologias de base de dados Oracle e PostgreSQL. Este facto permite que qual fosse a base de dados dos utilizadores poderiam se utilizar sem nenhum problema. O outro objetivo é que fosse possível mostrar o resultado dos documentos por ranking.

Este trabalho permitiu para eu aumentar os conhecimentos adquiridos ao longo do curso, bem como adquirir experiencia em diversas ferramentas. O fato de no curso existir alguma cadeira de programação, não permite ao estudante obter a prática e os conhecimentos necessários para se evoluir nesta área, sendo bastante importante a realização do estágio.

Um facto importante, trata-se de o mesmo ter sido realizado na ipBrick, que permitiu estar integrada no mundo real, observando e ter novos conhecimentos.

Neste momento a pesquisa esta a ser feita por ranking, mas seria interessante que o utilizador pudesse escolher o tipo de ordenação que deseja, descendente ou ascendente e onde quer que seja a ordenação, por data ou por título e que haja a opção do ranking, como um trabalho futuro.

Também seria interessante poder fazer a pesquisa dentro dos documentos e não nas informações dadas previamente. Assim a pesquisa seria mais exata já que ao pesquisar dentro do documento a informação seria mais exatas, e não ao pesquisar fora do documento já que o utilizador que introduziu as informações e o documento na base de dados pode se enganar e introduzir errado. Esse problema é mais uma falha humana. Por isso acho interessante pesquisar dentro dos documentos.

10. Bibliografia

Allan, J., Aslam, J., Belkin, N., Buckley, C., Callan, J., Croft, B., . . . ChengXiang, Z. (2002). Challenges in Information Retrieval and Language Modeling. *Report of a Workshop held at the Center for Intelligent Information Retrieva.* University of Massachusetts Amherst.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval.* Scotland: University of Glasgow.

Bartunov, O., & Sigaev, T. (s.d.). *Full-Text Search in PostgreSQL.*

Crestani, F., Lalmas, M., Rijsbergen, C. J., & Campbell, I. (1998). "Is this Document Relevant ?...Probably": A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys, 30*, pp. 528-552.

Franceschet, M. (2011). Pagerank: Standing on the shoulders of giants. *Communications of the ACM*.

Fuhr, N., & Pfeifer, U. (1994). Probabilistic Information Retrieval as Combination of Abstraction, Inductive Learning and Probabilistic Assumptions. *12*, pp. 92-115.

Gerard, S., & McGill, M. (1983). *introduction to Modern Information Retrieval.* McGraw Hill.

Greengrass, E. (2000). Information Retrieval: A Survey.

Harman, D. (s.d.). *Chapter 14: Ranking Algorithms*. Obtido em 2 de Novembro de 2012, de http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/toc.htm

Korfhage, R. R. (1997). *Information Storage and Retrieval.* New York: Wiley Computer Publishing.

Kowalski, G. (1997). *Information Retrieval Systems Theory and Implementation.* USA: Kluwer Academic Publishers.

Lee, J. H. (1994). Properties of Extended Boolean Models in Information Retrieval. *ACM SIGIR*, 182-190.

Lesk, M. (s.d.). *IFLANET*. Obtido em 27 de Outubro de 2012, de The Seven Ages of Information Retrieval: http://archive.ifla.org/VI/5/op/udtop5/udtop5.htm

Nunes, M., & O'Neil, H. (2004). *Fundamental de UML.* Lisboa: FCA.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web.

Pedrosa, A. C., & Gama, S. M. (2004). *Introdução Computacional a Probabilidade e Estatística.* Porto: Porto Editora.

Ricarte, I. L., & Gromide, F. (2004). A Reference Software Model for Intelligent. *Enchacing the power of Internet - Studies in Fuzziness and Soft Computing*, 327-346.

Rijisbergen, C. (1979). *Information Retrieval.* Glasgow: University of Glasgow.

Robertson, S., & Jones, K. (1976). Relevance Weighting of Search Terms. *Journal of the American Society for Information Science, 27*, pp. 129-146.

Rocha, A. (2012). *Relatório de gestão documental - funcionamento do programa iPortalDoc.* Gaya. Obtido de http://paginas.ispgaya.pt/~assr/docs/relatorio_ipdoc.pdf

Rocha, A. (2012). *Relatório de soluções de pesquisas de palavras-chaves em base de dados.* GAYA.

Salton, G. (1968). *Automatic Information Organization and Retrieval.* New York: McGraw-

Hill.

Salton, G., & McGill, M. (1983). *Introduction to Modern Information Retrieval.* USA: McGraw-Hill.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Information Retrieval and Language Processing*, 613-620.

Saravanan. (s.d.). *Tutorial Basics of Manual Testing*. Obtido em 2 de Outubro de 2012, de c-sharpcorner: http://www.c-sharpcorner.com/UploadFile/51e7af/basics-of-manual-testing/

Serrão, C., & Marques, J. (2007). *Programação com PHP 5.* Lisboa: FCA.

Sistemas de Gestión Documental. (s.d.). Obtido em 26 de Novembro de 2012, de Sistemas de Gestión Documental: www.uv.es/~rvcirilo/SGD/Tema1.ppt

Smith, E. (1993). On the shoulders of giants: from to Shannon to Taube: the origins of computerized information from the mid-19th century to the present. *Information Technology and Libraries*, pp. 217-226.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images* (2º Edição ed.). San Francisco: Morgan Kaufamann.

11. Anexos

Anexo A: Caso de Teste

Identificador ipDoc-CT1 Pesquisa por termos com o identificador do registo RT1

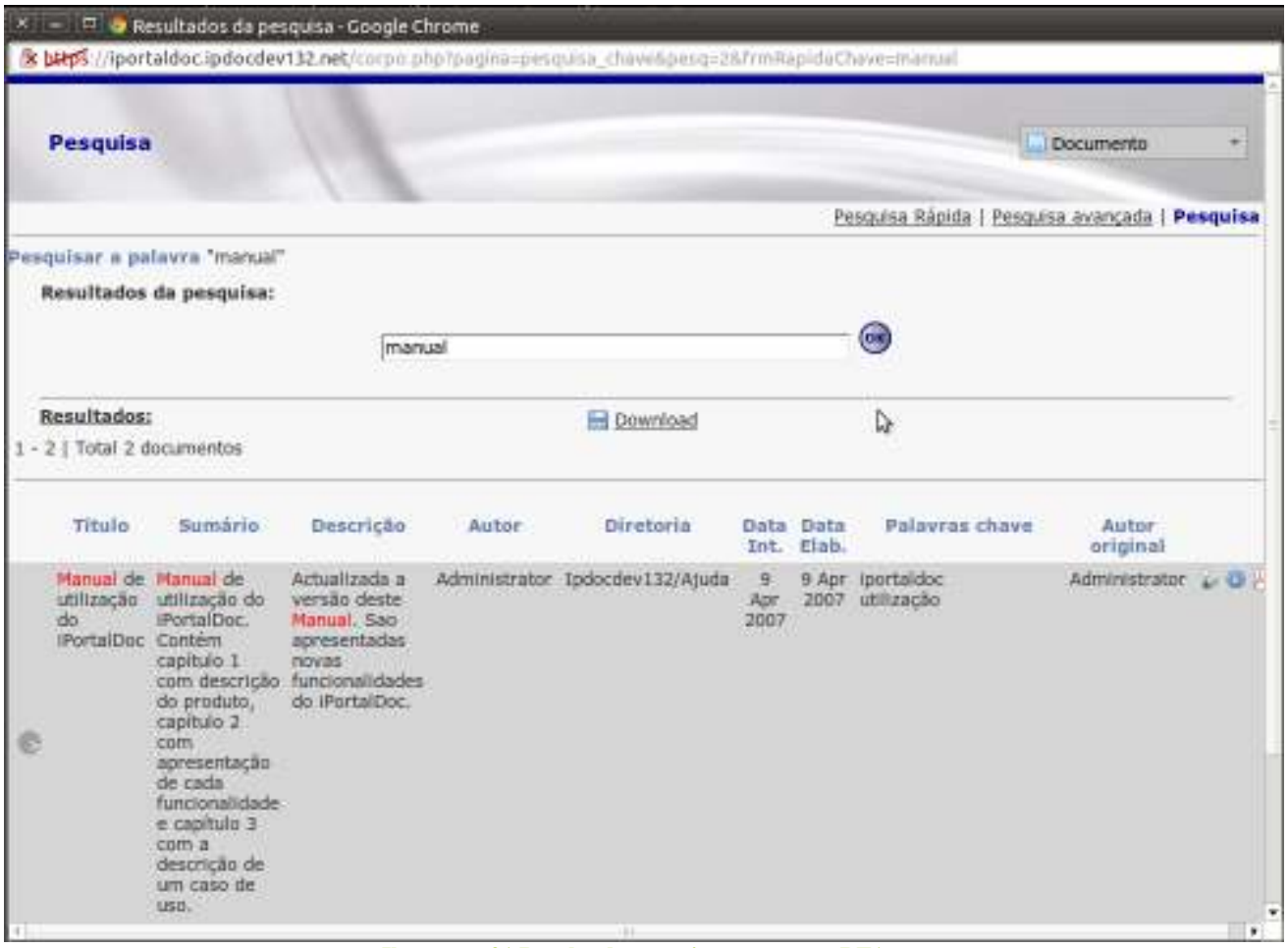

Ilustração 21 Resulta da pesquisa por termos RT1

Identificador ipDoc-CT2 Pesquisa por frases com o identificado com o identificador do registo RT2.

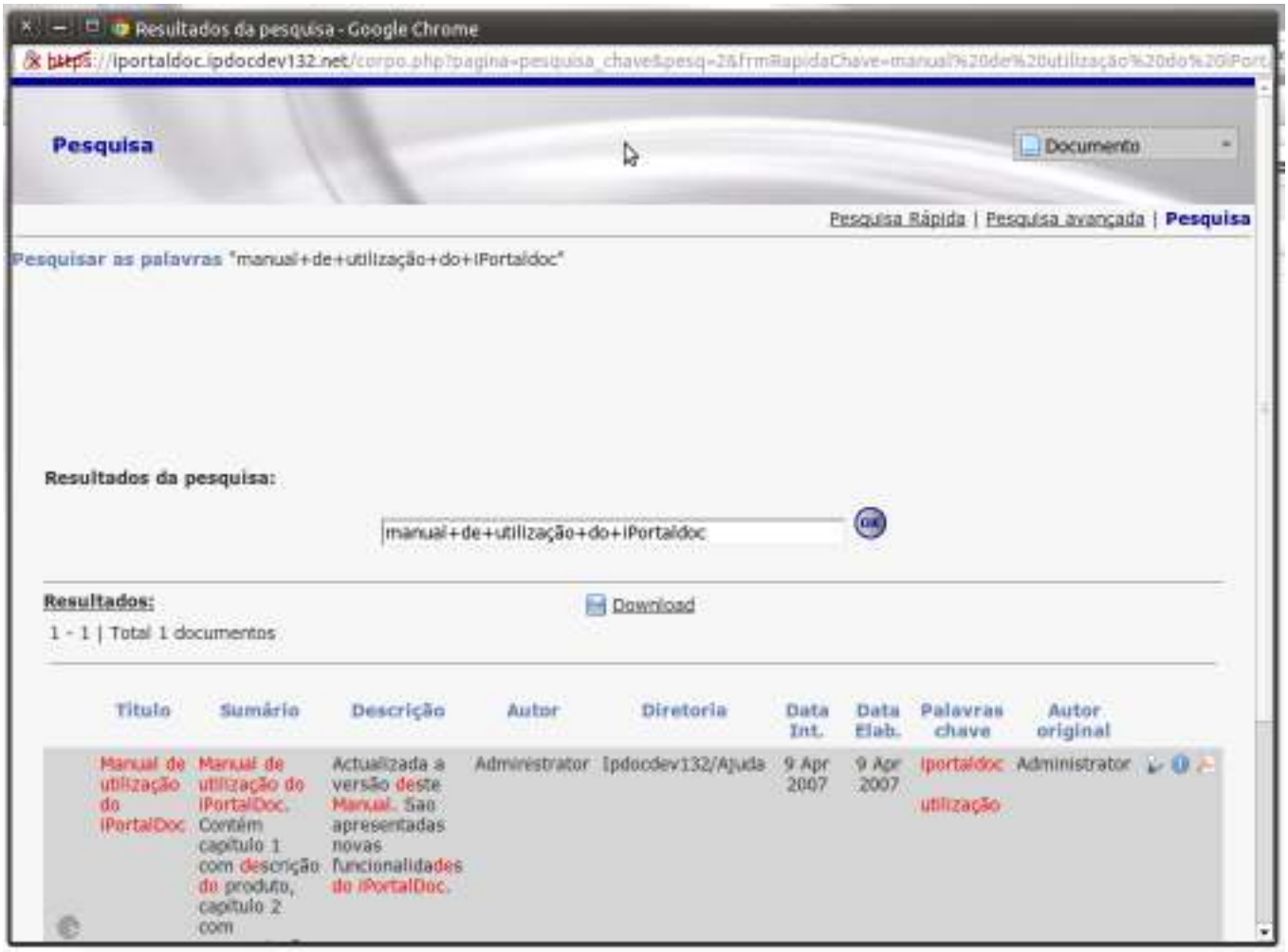

Ilustração 22 RT2 Pesquisa por frases

Identificador ipDoc-CT3 Pesquisa por termos com operadores boolaeanos (& and e | or) com o identificador do registo RT3.

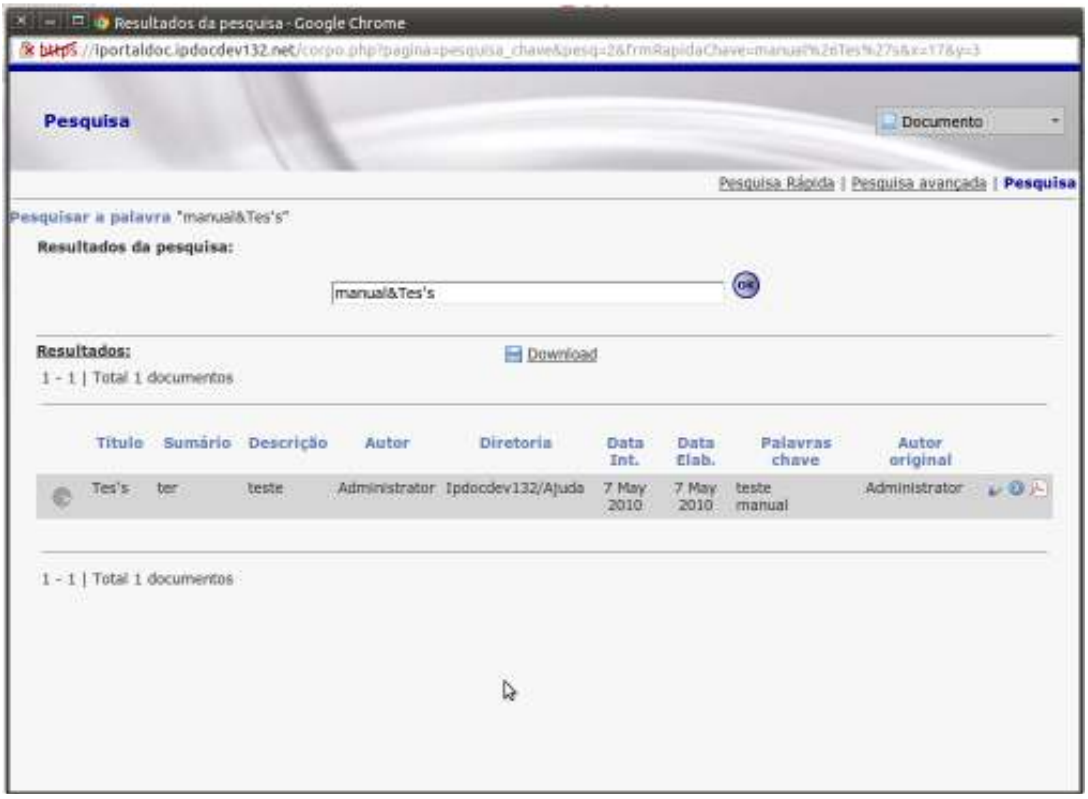

Ilustração 23 RT3 Pesquisa por termos com operador booleano & (and)

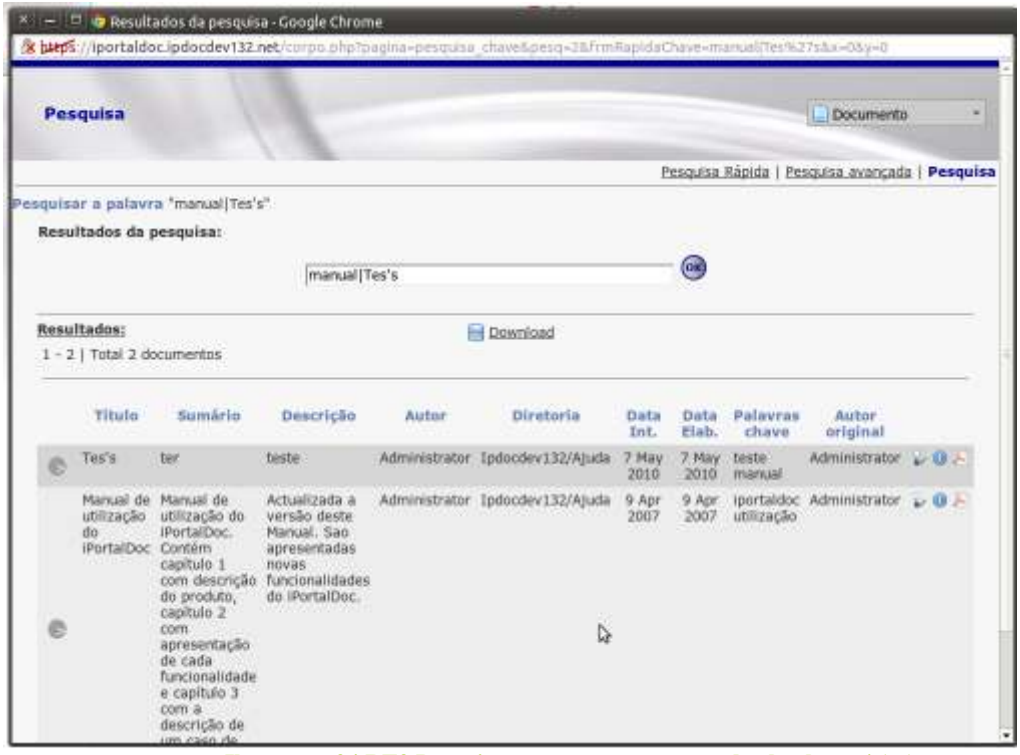

Ilustração 24 RT3 Pesquisa por termo com operador booleano | (or)

Identificador ipDoc-CT4 Pesquisa por termos com casos especiais com o identificador do registo RT4.

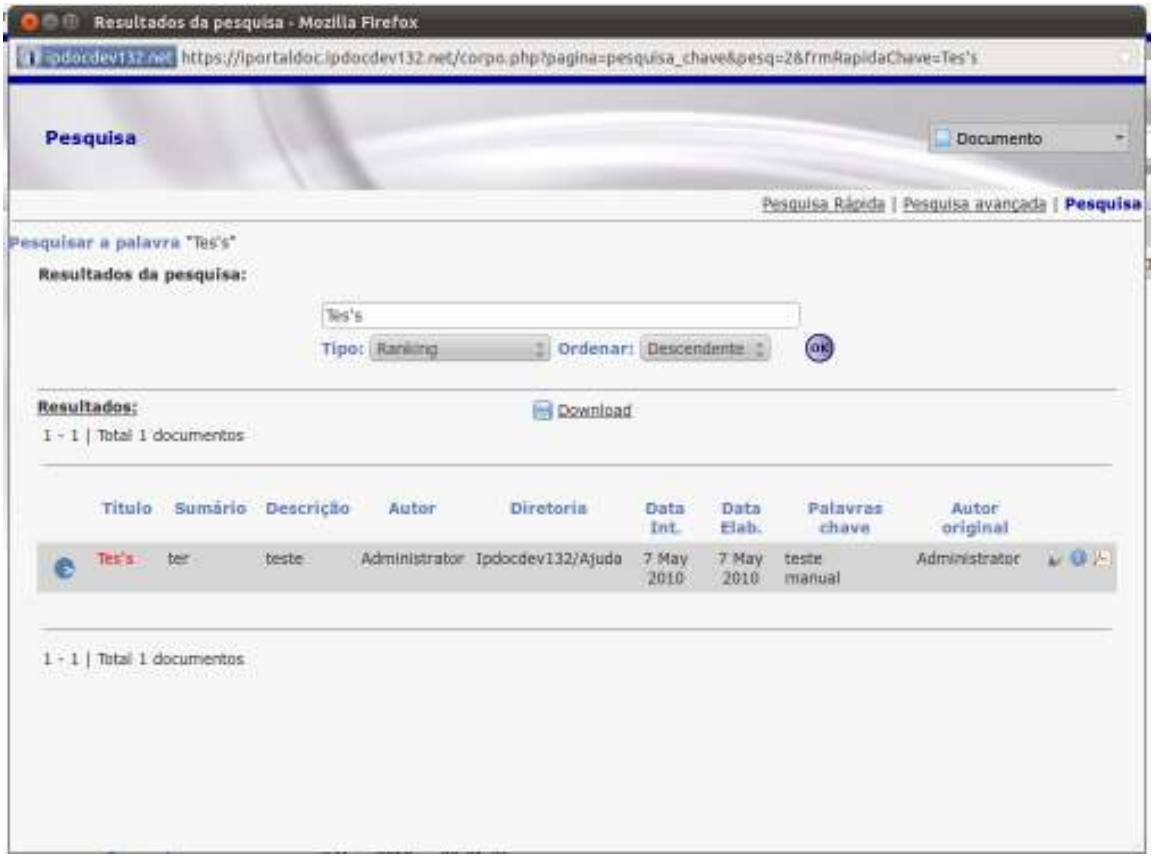

Ilustração 25 RT4 Pesquisa por termo com casos especiais

Identificador ipDoc-CT6 Escrever na base de dados com o identificador do registo RT5

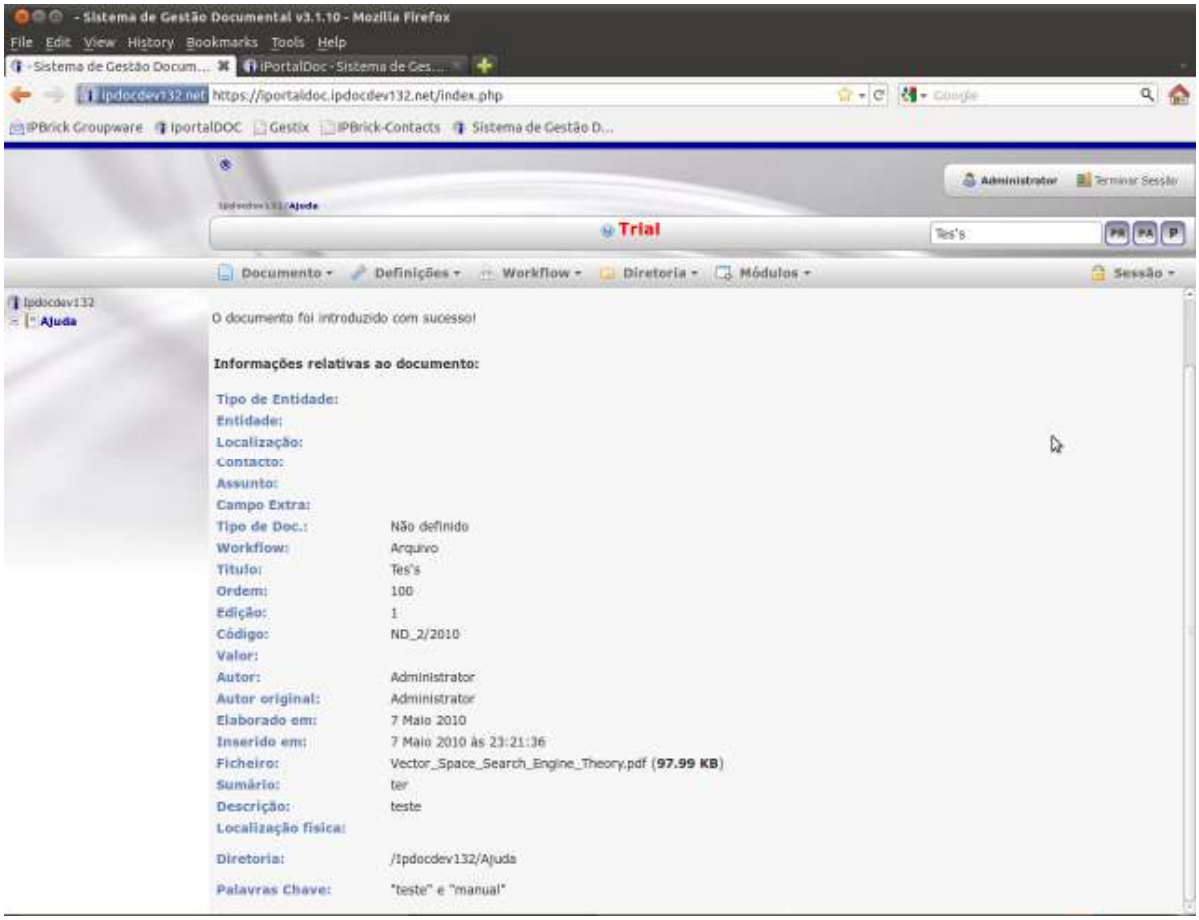

Ilustração 26 RT5 Escrever na base de dados e RT6 Atualizar o documento na base dados